Cascade Learning by Optimally Partitioning

Yanwei Pang, *Senior Member, IEEE,* Jiale Cao, and Xuelong Li, *Fellow, IEEE,*

***Abstract*—Cascaded AdaBoost classifier is a well-known efficient object detection algorithm. The cascade structure has many parameters to be determined. Most of existing cascade learning algorithms are designed by assigning detection rate and false positive rate to each stage either dynamically or statically. Their objective functions are not directly related to minimum computation cost. These algorithms are not guaranteed to have optimal solution in the sense of minimizing computation cost. On the assumption that a strong classifier is given, in this paper we propose an optimal cascade learning algorithm (we call it iCascade) which iteratively partitions the strong classifiers into two parts until predefined number of stages are generated. iCascade searches the optimal number r_i of weak classifiers of each stage i by directly minimizing the computation cost of the cascade. Theorems are provided to guarantee the existence of the unique optimal solution. Theorems are also given for the proposed efficient algorithm of searching optimal parameters r_i. Once a new stage is added, the parameter r_i for each stage decreases gradually as iteration proceeds, which we call decreasing phenomenon. Moreover, with the goal of minimizing computation cost, we develop an effective algorithm for setting the optimal threshold of each stage classifier. In addition, we prove in theory why more new weak classifiers are required compared to the last stage. Experimental results on face detection demonstrate the effectiveness and efficiency of the proposed algorithm.**

***Index Terms*—AdaBoost, cascade learning, classifier design, object detection.**

I. Introduction

ROBUST and real-time object detection is a key problem in computer vision tasks such as vision-based Human Computer Interaction (HCI), video surveillance and biometrics. Robustness of an object detection system is mainly governed by the robustness of extracted features and the generalization ability of employed classifiers. The detection efficiency is determined by the types of features, the manner of the features to be extracted, and the structure of the classifiers [1], [4], [5]. For example, it is well known that the features can be computed by the trick of integral image, which is suitable for efficient object detection. However, the structure of classifiers is also important for efficient object detection. For example, AdaBoost classifiers with cascade structure have greatly contributed to real-time face detection [10], [32], [30], [31], human detection [7], [25], [26], [28], [29], [2], [3], etc. With cascade structure, a large fraction of sub-windows can be rejected at early stages with a small number of weak classifiers. Only the sub-windows of true positives and those similar to true positives can arrive at later stages. However, how to design an optimal cascade structure is an open problem which is the focus of this paper.

Cascade learning is the process of determining the parameters of a cascade in order to improve the efficiency of AdaBoost classifier. The cascade parameters mainly include the number of stages, the number of weak classifiers in each stage, and the threshold for each stage. However, most of existing cascade learning methods are not directly formulated as a constrained optimization problem. Though more efficient than the non-cascade one, they are not guaranteed to be the best in the sense of maximizing detection efficiency under acceptable constraints. Usually, there are many hand-crafted parameters which are chosen according to one's our intuition and experience. The performance of the cascade AdaBoost relies on one's insight into the cascade structure. As Saberian and Vasconcelos mentioned [11], the design of a good cascade can take up several weeks. In addition, some useful intuitions are not justified in theory.

To overcome the above problems, we formulate cascade learning as a process of learning the parameters of a cascade by minimizing the computation cost with some certain constraints.

In summary, the contributions and characteristics of the paper are as follows.

1) We transform the strong classifier of regular AdaBoost into an optimal cascade classifier. That is, the result of regular AdaBoost is the input of our cascade learning algorithm. In the sense of detection rate and rejection rate, we use cascade AdaBoost to approach its non-cascade one (i.e., regular AdaBoost) with minimum computation cost.
2) The objective function of our method is just the computation cost of a cascade. In contrast, most of the existing algorithms are designed by empirically assigning detection rate and false positive rate to each stage either dynamically or statically. Existence and uniqueness of the optimal solution are analytically proved.
3) To design a one-stage cascade structure, we propose to partition the strong classifier $H(\mathbf{x})$, a combination of weak classifiers $h_1, ..., h_T$, into left part $H_L(\mathbf{x}, r_1)$ and right part $H_R(\mathbf{x}, r_1)$ at partition point r_1 (see Algorithm 1 and Fig. 1). The optimal partition point r_1 is found by minimizing the objective function $f_1(r)$ which stands for the computation cost of the cascade classifier. We theoretically (i.e., Theorem 1) prove that $f_1(r)$ exists a unique solution. Moreover, we give a theorem (i.e., Theorem 2) that gives a rough estimation of the optimal solution.
4) To design a two-stage cascade structure, we propose to further partition right classifier $H_R(\mathbf{x}, r_1)$ into two parts

Y. Pang and J. Cao are with the School of Electronic Information Engineering, Tianjin University, Tianjin 300072, China. e-mails: {pyw,connor}@tju.edu.cn

X. Li is with the Center for OPTical IMagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, Shaanxi, P. R. China. e-mail: xuelong_li@opt.ac.cn.

at partition point r_2. The partition iteratively continues (see Fig. 4). This algorithm is not globally optimal if r_1 is fixed while r_2 is considered as a variable. To obtain global optimization, we further jointly model the computation cost $f(r_1, r_2)$ with variables both r_1 and r_2. We prove that $f(r_1, r_2)$ has a unique minimum solution (see Theorem 7). An iterative optimization algorithm (i.e., Algorithm 2) is proposed to find the optimal solution. Theoretical analysis (i.e., Theorems 9-12) is given that r_1 decreases in each iteration where r_2 is fixed and r_2 decreases in each iteration where r_1 is fixed. We call it decreasing phenomenon. Such globally optimal two-stage cascade learning algorithm can be easily generalized to multi-stage one (i.e., Algorithm 3).

5) Moreover, we contribute to learning the optimal threshold t_i of each stage classifier for minimizing computation cost f_S of the cascaded classifier. We prove that the computation cost decreases with the stage threshold t_i (i.e., Theorem 13). Based on this theorem, we develop an effective threshold learning algorithm (i.e., Algorithm 4) whose core is properly decreasing t_i. Though this algorithm is not globally optimal, it is very effective. We call the proposed algorithm (i.e., Algorithm 4 and the procedure in Fig. 10) iCascade.
6) We prove in theory why more new weak classifiers are required compared to the previous stage (i.e., Theorem 5). In addition, we also theoretically prove why cascade AdaBoost is more efficient than its non-cascade one. Though the results and phenomena can be intuitively understood, we are the first to theoretically justify them to be the best of our knowledge.

II. Related Work

This section briefly reviews some existing work related to cascade leaning.

Most of existing cascade learning algorithms can be called DF-guided (where "DF" stands for Detection rate and False positive rate) method pioneered by Viola and Jones [8]. In the learning step, DF-guided method selects weak classifiers step by step until predefined minimum acceptable detection rate and maximum acceptable false positive rate are both satisfied. We call this method VJCascade [8] .

Variants of VJCascade have been proposed to select and organize weak classifiers. BoostChain [17] improves VJCascade by reusing the ensemble score from previous stages to enhance current stage. Brubaker et al. [10] called such a technique BoostChain recycling. Similar to BoostChain, SoftCascade also allows for monotonic accumulation of information as the classifier is evaluated [16]. In Multi-exit AdaBoost [22], node classifier also shares overlapping sets of weak classifiers. FloatBoost [18] as well as Boost-Chain uses DF-guided strategy to design the cascade. But different from VJCascade, FloatBoost uses backtrack mechanism to eliminate the less useful or even detrimental weak classifiers. Wu et al. [20] employed Forward Feature Selection (FFS) algorithm to greedily select features. Wang et al. [6] developed an asymmetric learning algorithm for both feature selection and ensemble classifier learning. FisherBoost [9] uses column generation technique to implement totally-corrective boosting algorithm. To decrease the training burden caused by the large number of negative samples and over-complete features (e.g., Haar-like features), some algorithms use only a random subset of the feature pool [10], [16].

Endeavor has also been devoted to adjust the thresholds of stages of a cascade structure which is also called the thresholds of node classifiers. On the assumption that a full cascade has been trained by VJCascade algorithm, Luo [19] proposed to jointly optimize the setting of the thresholding parameters of all the node classifiers within the cascade. Waldboost algorithm utilizes an adaptation of Wald's sequential probability ratio test to set stage thresholds [24]. Brubaker et al. proposed a linear program algorithm to select weak classifiers and threshold of a node classifier [10], [21].

Though most of existing methods are DF-guided, computation-cost guided (i.e., CC-guided) methods were also developed. Chen and Yuille [23] gave a criterion for designing a time-efficient cascade that explicitly takes into account the time complexity of tests including the time for pre-processing. They designed a greedy algorithm to minimize the criterion. But each stage in this method is constrained to detect all positive examples, which leads it to miss opportunity to improve detection efficiency [13]. The loss function of Cronus cascade learning algorithm is a tradeoff between accuracy (training error) and computation cost [13]. CSTC (i.e., Cost-Sensitive Tree of Classifiers) combines regularized training error and computation cost into a loss function [15]. Compared to VJCascade-like method, CSTC is suitable for balanced classes and specialized features [15].

In contrast to the above methods, the objective function (i.e., loss function) of our method is just the computation cost and the detection accuracy can be naturally guaranteed. In addition, global solution instead of local one can be obtained in our method.

III. Proposed Method: One-Stage Cascade

The goal of cascade learning is to lean a cascade structure in order to correctly reject negative sub-windows and accept positive sub-windows as fast as possible. Generally, the cascade structure is determined by the number of stages and the number of weak classifiers in each stage.

Most of existing methods design or learn the cascade structure by assigning minimum acceptable detection rate and maximum acceptable false positive rate for each stage. In this paper, we propose a novel cascade learning method in which it is not necessary to assign such acceptable detection rates and false positive rates. Instead, we learn the parameters of a cascade by directly minimizing the computation cost.

In this section, we describe the proposed one-stage cascade learning algorithm which is the foundation of our multi-stage cascade learning algorithm.

A. Testing Stage

In our method, cascade AdaBoost is considered as an estimation of regular AdaBoost. A good cascade structure can

achieve the same detection accuracy as AdaBoost with small computation cost. Therefore, we begin with describing the form of the strong classifier of regular AdaBoost.

Let $H(\mathbf{x})$ be the strong classifier obtained by an AdaBoost algorithm. The strong classifier $H(\mathbf{x})$ is composed of T weak classifier $h_i(\mathbf{x}) \in \{1, -1\}$ with weights α_i:

$$H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x}). \tag{1}$$

Generally, the weights of the weak classifiers satisfy

$$\alpha_1 > \alpha_2 > \cdots > \alpha_T > 0, \tag{2}$$

and

$$\sum_{i=1}^{T} \alpha_i = 1. \tag{3}$$

Let $l(\mathbf{x}) \in \{1, -1\}$ be the class label of a feature vector. The decision rule of the strong classifier $H(\mathbf{x})$ is:

$$l(\mathbf{x}) = \begin{cases} 1, & \text{if } H(\mathbf{x}) = \sum\limits_{i=1}^{T} \alpha_i h_i(\mathbf{x}) > t, \\ -1, & \text{if } H(\mathbf{x}) = \sum\limits_{i=1}^{T} \alpha_i h_i(\mathbf{x}) \le t, \end{cases} \tag{4}$$

where t is a threshold balancing the detection rate and false positive rate.

In one-stage cascade structure, there is only one stage in which a small number (i.e., r) of weak classifiers are combined for classification. The core of the proposed one-stage cascade is to determine an optimal r which divides the strong classifier $H(\mathbf{x})$ into left part $H_L(\mathbf{x})$ and right part $H_R(\mathbf{x})$:

$$H(\mathbf{x}) = H_L(\mathbf{x}) + H_R(\mathbf{x}), \tag{5}$$

$$H_L(\mathbf{x}) = \sum_{i=1}^{r} \alpha_i h_i(\mathbf{x}), \tag{6}$$

$$H_R(\mathbf{x}) = \sum_{i=r+1}^{T} \alpha_i h_i(\mathbf{x}). \tag{7}$$

To reject true negative sub-windows with less computation cost, we propose to use the maximum of $H_R(\mathbf{x})$ to approximate the value of $H_R(\mathbf{x})$:

$$\max H_R(\mathbf{x}) = \max_{\mathbf{x}} \left(\sum_{i=r+1}^{T} \alpha_i h(\mathbf{x}) \right) = \sum_{i=r+1}^{T} \alpha_i. \tag{8}$$

We denote the maximum by $\max H_R(\mathbf{x})$. With $\max H_R(\mathbf{x})$, it is guaranteed that all the true negative sub-windows can be correctly rejected if the following inequality holds:

$$H_L(\mathbf{x}, r) + \max H_R(\mathbf{x}, r) \le t. \tag{9}$$

That is, some sub-windows can be rejected by using merely $H_L(\mathbf{x})$ and $\max H_R(\mathbf{x})$ instead of both $H_L(\mathbf{x})$ and $H_R(\mathbf{x})$. Consequently, the computation cost is significantly reduced.

The rest sub-windows not satisfying (9) have to be classified using both $H_L(\mathbf{x})$ and $H_R(\mathbf{x})$ (i.e., the strong classifier). If the sum of $H_L(\mathbf{x})$ and $H_R(\mathbf{x})$ is not larger than t, i.e.,

$$H_L(\mathbf{x}, r) + H_R(\mathbf{x}, r) = H(\mathbf{x}) \le t, \tag{10}$$

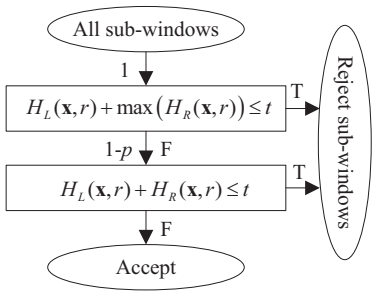


Fig. 1. Proposed method: one stage cascade AdaBoost with a given r.

Algorithm 1 One-stage cascade

1: if $H_L(\mathbf{x}, r) + \max H_R(\mathbf{x}, r) \le t$,
2: then $l(\mathbf{x}) = -1$,
3: else (i.e., $H_L(\mathbf{x}, r) + \max H_R(\mathbf{x}, r) > t$)
4: if $H_L(\mathbf{x}, r) + H_R(\mathbf{x}, r) \le t$
5: $l(\mathrm{x}) = -1$,
6: else (i.e., $H_L(\mathbf{x}, r) + H_R(\mathbf{x}, r) > t$)
7: $l(\mathbf{x}) = 1$.

then the sub-window corresponding to the feature vector **x** can be finally classified as negative sub-window. Otherwise (i.e., the sum is larger than t), it is classified as positive sub-window. The algorithm of one-stage cascade is given in Algorithm 1. Equivalently, the flow-chart is shown in Fig. 1. Note that $t - \max H_R(\mathbf{x}, r)$ can be viewed as the threshold for $H_L(\mathbf{x}, r)$. The issue of how to set the threshold is addressed in Section V.C.

B. Training Stage: How to Select an Optimal r

In Fig. 1, it is assumed that r and $\max H_R(\mathbf{x}, r)$ are given. In this sub-section, we describe how to choose an optimal r. $\max H_R(\mathbf{x}, r)$ can be easily computed from training samples once r is given. In the training stage of cascade learning, it is assumed that the strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$ is obtained by a regular AdaBoost Algorithm.

Given r, a p fraction of true negative sub-windows can be rejected by using left classifier $H_L(\mathbf{x})$ (i.e., (9)). The fraction p is called rejection rate and defined by:

$$p(\mathbf{x}) = \frac{\sum\limits_{\mathbf{x}} I(\sum\limits_{i=1}^{r} \alpha_i h_i(\mathbf{x}) + \max H_R(\mathbf{x}, r) \le t)}{\sum\limits_{\mathbf{x}} I(l(\mathbf{x}) == -1)}, \tag{11}$$

where $I(condition)$ is 1 if the condition is satisfied and 0 otherwise. $\sum_{\mathbf{x}} I(l(\mathbf{x}) == -1)$ is the number of all true negative sub-windows. Eq. (11) shows that p is dependent on r.

Obviously, the fraction of true negative sub-windows classified by using both left and right classifiers is $1 - p$. The criterion for choosing r is to minimize the overall computation cost f consisting of the cost f^L of computing $H_L(\mathbf{x}, r)$ in (9) and the cost f^R of computing both $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$ in (10).

Suppose that all the weak classifiers have the same computation complexity. Then the computation cost is determined by

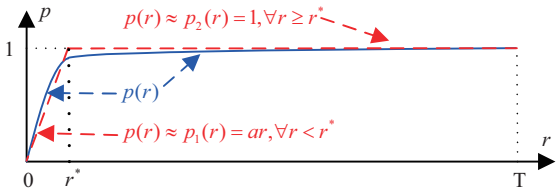


Fig. 2. A representative form of function $p(r)$ which can be simplified as combination of two linear functions: $p_1(r) = ar$ with $r < r^*$ and $p_2(r) = 1$ with $r \geq r^*$.

the number of weak classifiers. A fact is that f_1^L grows with p and r:

$$f_1^L(r, p) = p(r + c), \tag{12}$$

and f_1^R grows with $1 - p$ and T:

$$f_1^R(r, p) = (1 - p)(T + 2c). \tag{13}$$

In (12) and (13), c is the computational cost of checking either inequality (9) or inequality (10) holds. Usually, the computation cost C of a weak classifier is bigger than c. Let $C = 1$, then $c < 1$. Note that c is not involved in computing $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$.

The goal is to minimize the following object function:

$$f_1(r, p) = f_1^L(r, p) + f_1^R(r, p) = p(r + c) + (1 - p)(T + 2c). \tag{14}$$

To solve this optimization problem, it is necessary to reveal the relationship between r and p. The parameters r and p are correlated and the correlation can be expressed as a function $p(r, \max H_R(\mathbf{x}, r))$.

As $\max(H_R(\mathbf{x}, r))$ (its upper bound is $\sum_{i=r+1}^{T} \alpha_i$) decreases with r, a larger number of negative sub-windows will be rejected by (9). It is straightforward that the fraction p of negative sub-windows satisfying (9) grows with r. Experimental results also show that p monotonically increases with r. The relationship between p and r is nonlinear. Fig. 2 illustrates a typical trend that how p varies with r. It can be seen that p grows quickly from 0 to the value (e.g., 0.99) close to 1 when r changes from 1 to a small value r^* (e.g., 10). But p becomes stable when r is larger than r^*. The reason is that the first r^* weak classifiers h_i with larger weights α_i play much more important role than the rest weak classifiers.

Mathematically, r^* is defined as the minimum r which satisfies $p \approx 1$ or equivalently $1 - p(r) \leq \varepsilon$ with ε being a small number (e.g., 0.01):

$$r^* = \arg\min_r \{r | 1 - p(r) \leq \varepsilon\}. \tag{15}$$

We call r^* the saturation point of $p(r)$.

Though $p(r)$ is in fact a high-order curve, it can be well modelled by combining of two linear functions: $p_1(r) = ar$ with $r < r^*$ and $p_2(r) = 1$ with $r \geq r^*$(see Fig. 2). As T is a large number, then $r^* \ll T$.

It is reasonably assumed that the function $p(r)$ satisfies the fowllowing conditions:

$$p(r_1) < p(r_2), \ if \ r_1 < r_2, \tag{16}$$
$$p'(r_1) > p'(r_2) \geq 0, \ if \ r_1 < r_2, \tag{17}$$
$$p(T) = 1, \tag{18}$$
$$p(0) = 0, \tag{19}$$
$$p'(T) = 0, \tag{20}$$
$$p'(0) \gg 0, \tag{21}$$
$$p'(1) \gg 0. \tag{22}$$

(16) states the monotonicity of $p(r)$. (17) tells that the slope of $p(r)$ decreases with r. (20) shows that the slope is zero at $r = T$ while it is extremely large at $r = 1$. It is noted that (16)-(22) will be used as assuption of the theorems of the proposed methods.

According to Fig. 2, $p(r)$ has the following properties:

$$r^* \ll T, \ as \ T \ is \ a \ large \ number, \tag{23}$$
$$p(r) \approx p_1(r) = ar, \ if \ r < r^*, \tag{24}$$
$$p(r) \approx p_2(r) = 1, \ if \ r \geq r^*, \tag{25}$$

which will be used as assumption of Theorem 2.

After each pair of (r, p) are known, the value of f_1 can be obtained. Theorem 1 tells that there exists a unique minimization solution.

Theorem 1. *$f_1(r) = p(r)(r + c) + (1 - p(r))(T + 2c)$ has a unique minimum solution r_1. Moreover, $f_1(r)$ monotonically decreases with r until $r = r_1$ and then increases with r.*

Proof. We first prove the existence of the minimum solution and then give the evidence of the uniqueness of the minimum solution.

Existence:

$\because f_1(r) = p(r)(r + c) + (1 - p(r))(T + 2c)$,

$\therefore f_1'(r) = p'(r)(r - T - c) + p(r)$.

Consider the value of the derivative $f_1'(r)$ when r approaches 0:

$$\lim_{r \to 0} f_1'(r) = f_1'(0) = p'(0)(0 - c - T) + p(0). \tag{26}$$

Because $p(0) = 0$ (i.e., (19)) and $p'(0) \gg 0$ (i.e., (21)). Therefore, it holds:

$$\lim_{r \to 0} f_1'(r) = f_1'(0) = -p'(0)(T + c) < 0. \tag{27}$$

Now consider the value of the derivative $f_1'(r)$ when r approaches T:

$$\lim_{r \to T} f_1'(r) = f_1'(T) = p(T) - p'(T)c \approx 1 - 0 > 0. \tag{28}$$

Because $\lim_{r \to 0} f_1'(r) < 0$, $\lim_{r \to T} f_1'(r) > 0$ and $f_1'(r)$ is continuous function, it must exist a $r_1 \in [1, T]$ such that $f_1'(r_1) = 0$. The r_1 is at least a local minimum, which shall be the global minimum if the local minimum is unique.

Uniqueness (Proof by contradiction):

Suppose that there are two local minimums r_1 and r_2 with $r_1 < r_2$. Then it holds that $f_1'(r_1) - f_1'(r_2) = 0$.

Now investigate the value of $f_1'(r_1) - f_1'(r_2) = [p(r_1) - p(r_2)] - [p'(r_1)(T + c - r_1) - p'(r_2)(T + c - r_2)]$ if $r_1 < r_2$

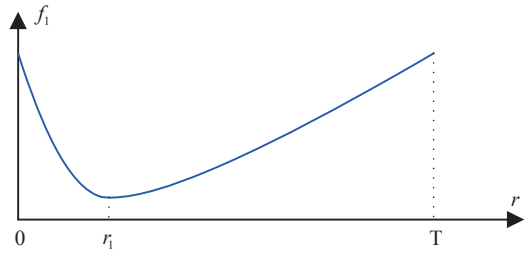


Fig. 3. A representative form of $f_1(r)$

is true.
$\because r_1 < r_2$,
$\therefore p'(r_1) > p'(r_2) > 0,\ T + c - r_1 > T + c - r_2 > 0,$
$p(r_1) < p(r_2)$,
$\therefore f'(r_1) - f'(r_2) < 0.$
This contradicts $f'(r_1) - f'(r_2) = 0$, Therefore, $r_1 < r_2$ is wrong.

Similarly, we can prove that $r_1 > r_2$ is wrong. Consequently, $r_1 = r_2$ is true, meaning a unique solution. □

Fig. 3 shows a representative form of $f_1(r)$, it has a unique minimum solution.

Theorem 2. *Let r^* be a saturation point of $p(r)$ (see (15)) and assume that $p(r)$ can be modelled by combining $p_1(r) = ar$ where $r < r^*$ with $p_2(r) = 1$ where $r \geq r^*$(see Fig. 2 for illustration). Then the saturation point r^* is the optimal minimum solution $r_1 = \arg\min_r f_1(r)$.*

Proof. Note that $f_1^L(r) = p(r)(r + c)$, $f_1^R(r) = (1 - p(r))(T + 2c)$, and $f_1(r) = f_1^L(r, p) + f_1^R(r, p) = p(r)(r + c) + (1 - p(r))(T + 2c)$.

Case 1: For $r^* \leq r \leq T$, because $p_2(r) = 1$ and $1 - p_2(r) = 0$, so we have $f_1^L(r) = p_2(r)(r + c) = r + c$, $f_1^R(r) = 0$, and hence $f_1(r) = r + c$. Therefore, the optimal solution r_R^* for $r \geq r^*$ is r^* itself. That is,

$$r_R^* = \arg \min_{r^* \leq r \leq T} f_1(r) = r^*. \tag{29}$$

Case 2: For $0 \leq r \leq r^*$, because $p(r) \approx p_1(r) = ar$, we have:

$$f_1^L(r) = p_1(r)(r + c) = ar(r + c),$$
$$f_1^R(r) = (1 - p(r))(T + 2c) = (1 - ar)(T + 2c),$$
$$f_1(r) = f_1^L(r, p) + f_1^R(r, p) = ar^2 - a(T + c)r + (T + 2c),$$
$$f_1' = 2ar - a(T + c) = 0 \Rightarrow \widetilde{r} = \arg\min_r f_1(r) = (T + c)/2.$$

Because $r^* < \widetilde{r}$, $f_1(0) = T + 2c$, and $f_1(r)$ monotonically decreases with r when $r < \widetilde{r}$, the minimum value r_L^* of $f_1(r)$ in the range of $0 < r \leq r^*$ is r^*. That is,

$$r_L^* = \arg \min_{0 \leq r \leq r^*} f_1(r) = r^*. \tag{30}$$

It is observed from (29) and (30) that the minimum solutions for $0 \leq r \leq r^*$ and $r^* \leq r \leq T$ are identical to r^*. Consequently, $r^* = \arg \min_{0 \leq r \leq T} f_1(r)$.

Therefore, optimal minimum solution $r_1 = \arg\min_r f_1(r)$ is r^*. □

IV. Proposed Method: Local-Minimum Based Multi-Stage Cascade

In this section, we extend one-stage cascade learning to multi-stage cascade learning.

A. Testing Stage

From (5), one can see that one-stage cascade is obtained by splitting $H(\mathbf{x})$ into $H_L(\mathbf{x}, r_1)$ and $H_R(\mathbf{x}, r_1)$ where r_1 is the optimal r (i.e., $r_1 = \arg\min_r f_1(r)$). We add a superscript "1" to H_L and H_R so that one explicitly knows that $H_L^1(\mathbf{x}, r_1)$ and $H_R^1(\mathbf{x}, r_1)$ correspond to stage 1. Multi-stage cascade is obtained by iteratively splitting the right classifier H_R^i.

As shown in Fig. 4, the sub-windows not rejected by stage 1 are fed to stage 2. The second stage is obtained by further dividing the right classifier $H_R^1(\mathbf{x}, r_1)$ into two parts at the partition point r_2 ($r_2 > r_1$):

$$H_R^1(\mathbf{x}, r_1) = H_L^2(\mathbf{x}, r_2) + H_R^2(\mathbf{x}, r_2), \tag{31}$$

$$H_L^2(\mathbf{x}, r_2) = \sum_{i=r_1+1}^{r_2} \alpha_i h_i(\mathbf{x}), \tag{32}$$

$$H_R^2(\mathbf{x}, r_2) = \sum_{i=r_2+1}^{T} \alpha_i h_i(\mathbf{x}). \tag{33}$$

In stage 2, the sub-windows are rejected if the following inequality holds:

$$H_L^1(\mathbf{x}, r_1) + \left[H_L^2(\mathbf{x}, r_2) + \max\left(H_R^2(\mathbf{x}, r_2)\right)\right] \leq t. \tag{34}$$

(34) is equivalent to

$$H_L(\mathbf{x}, r_2) + \max\left(H_R(\mathbf{x}, r_2)\right) \leq t, \tag{35}$$

because

$$H_L(\mathbf{x}, r_2) = H_L^1(\mathbf{x}, r_1) + H_L^2(\mathbf{x}, r_2). \tag{36}$$

But (35) is more time-consuming than (34) because r_2 ($r_2 > r_1$) weak classifiers are used to compute $H_L(\mathbf{x}, r_2)$ in (35) whereas in (34) $H_L^1(\mathbf{x}, r_1)$ has been computed in stage 1 and $H_L^2(\mathbf{x}, r_2)$ can be efficiently computed using as small as $(r_2 - r_1)$ weak classifiers where $H_L^1(\mathbf{x}, r_1)$ can be reused in stage 2.

Analogously, the left classifier in stage $i - 1$ can be represented by the left and right classifiers in stage i:

$$H_R^{i-1}(\mathbf{x}, r_{i-1}) = H_L^i(\mathbf{x}, r_i) + H_R^i(\mathbf{x}, r_i). \tag{37}$$

The block diagram of the multi-stage cascade is shown in Fig. 4 where the rejection rate p_i is the ratio of sub-windows rejected in Stage i. In stage 1, p_1 fraction of sub-windows are directly rejected and $1 - p_1$ fraction of sub-windows are fed to stage 2. Among the $(1 - p_1)w$ sub-windows, p_2 fractions are rejected by stage 2 and $1 - p_2$ fractions are considered as positive-class candidates and therefore are fed to stage 3. This means that $(1-p_1)(1-p_2)$ fraction of total w sub-windows are to be classified by stage 3. Because p_i in stage i is dependant on p_{i-1} in stage $i - 1$, we explicitly express p_i as $p(r_i|r_{i-1})$

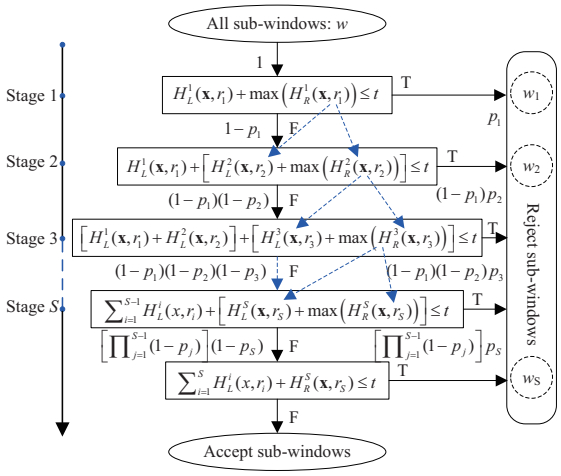


Fig. 4. Testing process of local-minimum based multi-stage cascade AdaBoost.

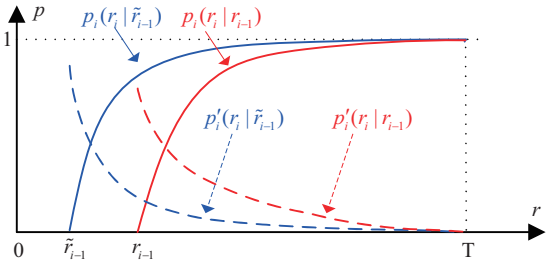


Fig. 5. The form of $p_i(r_i|r_{i-1})$ and its properties. If $\widetilde{r}_{i-1} < r_{i-1}$, then $p_i(r_i|\widetilde{r}_{i-1}) > p_i(r_i|r_{i-1})$ and $p_i'(r_i|\widetilde{r}_{i-1}) < p_i'(r_i|r_{i-1})$.

when necessary. Specifically, the rejection rate $p(r_i|r_{i-1})$ is defined as:

$$p(r_i|r_{i-1}) = \frac{I(\sum_{k=1}^{r_i} \alpha_k h_k(\mathbf{x}) + \max(H_R(\mathbf{x}, r_i)) \le t)}{I(\sum_{k=1}^{r_{i-1}} \alpha_k h_k(\mathbf{x}) + \max(H_R(\mathbf{x}, r_{i-1})) > t)}. \tag{38}$$

Fig. 5 shows two representative curves of $p(r_i|r_{i-1})$. The properties of $p(r_i|r_{i-1})$ are summarized as follows:

$$p'(r_i|r_{i-1}) \ge 0, \tag{39}$$

$$\lim_{r_i \to r_{i-1}} p(r_i|r_{i-1}) = 0, \tag{40}$$

$$\lim_{r_i \to T} p(r_i|r_{i-1}) = 1, \tag{41}$$

$$p(r_i|\widetilde{r}_{i-1}) > p(r_i|r_{i-1}), \ if \ \widetilde{r}_{i-1} < r_{i-1}, \tag{42}$$

$$p'(r_i|\widetilde{r}_{i-1}) < p'(r_i|r_{i-1}), \ if \ \widetilde{r}_{i-1} < r_{i-1}. \tag{43}$$

We give a theoretical guarantee (i.e., Theorem 3) that adding a stage results in reduction in computation cost if certain condition is satisfied.

Theorem 3. *Let $r_1, \ldots, r_S$ define an S stage cascade structure whose computation cost is $f_S(r_1, \ldots, r_S)$:*

$$f_S(r_1, \ldots, r_S) = \sum_{i=1}^{S} f_i^L(r_1, \ldots, r_i) + f_S^R(r_S), \tag{44}$$

where

$$f_i^L = \left[\prod_{j=1}^{i-1}(1 - p_j(r_j|r_{j-1}))\right] p_i(r_i|r_{i-1})(r_i + ic), \tag{45}$$

$$f_S^R = \left[\prod_{j=1}^{S}(1 - p_j(r_j|r_{j-1}))\right] (T + (S+1)c). \tag{46}$$

Let $r_1, \ldots, r_{S-1}$ define an $S-1$ stage cascade structure whose computation cost is $f_{S-1}(r_1, \ldots, r_{S-1})$:

$$f_{S-1}(r_1, \ldots, r_{S-1}) = \sum_{i=1}^{S-1} f_i^L(r_1, \ldots, r_i) + f_{S-1}^R(r_{S-1}). \tag{47}$$

If $p_S(r_S|r_{S-1}) > c/(T + c - r_S)$, then we have

$$f_{S-1}(r_1, \ldots, r_{S-1}) > f_S(r_1, \ldots, r_{S-1}, r_S). \tag{48}$$

Proof.

$$\begin{aligned}
&\because f_{S-1}(r_1, \ldots, r_{S-1}) - f_S(r_1, \ldots, r_{S-1}, r_S) \\
&= f_{S-1}^R(r_{S-1}) - f_S^L(r_1, \ldots, r_S) - f_S^R(r_S) \\
&= \left[\prod_{j=1}^{S-1}(1 - p_j)\right] [p_S(r_S|r_{S-1})(T + c - r_S) - c], \\
&\because 1 - p_j > 0, \ p_S > 0, \\
&\therefore \text{if } p_S(r_S|r_{S-1}) > c/(T + c - r_S), \text{then } f_{S-1} > f_S.
\end{aligned}$$

□

Note that if the computational cost c is omitted, then $f_{S-1} > f_S$ as long as $p_S(r_S|r_{S-1}) > 0$. In this case, it is optimal that each stage contains a new weak classifier (i.e., the case $S = T$, $r1 = 1, r2 = 2, ..., r_T = T$). But $c \neq 0$ in practice, it is necessary to let $S < T$ and find way to search the optimal values of $r_1, ..., r_S$.

B. Training Stage: How to Select Optimal r_i

Section IV.A describes the testing stage of the proposed cascade method. Now we describe the training stage of the proposed method which is closely related to Section III.B.

1) Existence and Uniqueness: Investigating Fig. 4, one can find that the cascade structure is completely determined once $r_1, \ldots, r_S$ are known. Therefore, the main task of the training stage is to find the optimal $r_1, \ldots, r_S$.

The r_1 in stage 1 is obtained by the method in Section III.B. Given r_1, we learn the best r_2:

$$r_2 = \arg\min_r f_2(r_1, r) = \arg\min_r f_2(r|r_1). \tag{49}$$

Similar to the proof of Theorem 1, it can be proved that $f_2(r|r_1)$ has a unique solution.

Generally, r_i is computed based on $r_1, \ldots, r_{i-1}$:

$$\begin{aligned}
r_i &= \arg\min_{r_{i-1} < r < T} f_i(r_1, \ldots, r_{i-1}, r) \\
&= \arg\min_{r_{i-1} < r < T} f_i(r|r_1, \ldots, r_{i-1}).
\end{aligned} \tag{50}$$

In (50), $f_i(r|r_1, \ldots, r_{i-1})$ is used to describe the assumption that $r_1, \ldots, r_{i-1}$ in the first $i-1$ stages are given. If $r_1, \ldots, r_{i-1}$ and $p(r_1), p(r_2|r_1), \ldots, p(r_{i-1}|r_{i-2})$ are known,

then $f_i(r|r_1,\ldots,r_{i-1})$ will be in the similar form as $f_1(r)$ (see (14)):

$$\begin{aligned}
&f_i(r_i|r_1,\ldots,r_{i-1})\\
&=\sum_{j=1}^{i} f_j^L(r_1,\ldots,r_j)+f_i^R(r_i)\\
&=\sum_{j=1}^{i-1} f_j^L(r_1,\ldots,r_j)+f_i^L(r_1,\ldots,r_i)+f_i^R(r_i)\\
&=\left[\sum_{j=1}^{i-1} f_j^L(r_1,\ldots,r_j)\right]\\
&+\left[\prod_{j=1}^{i-1}(1-p_j(r_j|r_{j-1}))\right]p_i(r_i|r_{i-1})(r_i+ic)\\
&+\left[\prod_{j=1}^{i-1}(1-p_j(r_j|r_{j-1}))\right](1-p_i(r_i|r_{i-1}))(T+(i+1)c).
\end{aligned} \tag{51}$$

Because the items in bracket in (51) are constant, so $f_i(r|r_1,\ldots,r_{i-1})$ is in the similar form as $f_1(r_1)$. Therefore, as a corollary of Theorem 1, we have the following theorem:

Theorem 4. $\min_r f_i(r_1,\ldots,r_{i-1},r)=\min_r f_i(r|r_1,\ldots,r_{i-1})$ *has a unique minimum solution* $r_i\in[r_{i-1},T]$. *Moreover,* $f_i(r|r_1,\ldots,r_{i-1})$ *monotonically decreases with* r *until* $r=r_i$ *and then increases with* r.

Theorem 4 implies that $f_i(r|r_1,\ldots,r_{i-1})$ has the similar form as the curve in Fig. 3.

2) Efficient Search: The search range of r_i is (r_{i-1},T). However, because $f_i(r|r_1,\ldots,r_{i-1})$ monotonically decreases with r until $r=r_i$ and then increases with r, to find the unique minimum solution one can increase r from r_{i-1} with a small step and stop at the value once $f_i(r|r_1,\ldots,r_{i-1})$ no longer decreases. Therefore, the practical range is less than (r_{i-1},T).

The search range can be further reduced according to the following increasing phenomenon.

Theorem 5. *If* $r_i=\arg\lim_{r_{i-1}<r<T} f_i(r|r_1,\ldots,r_{i-1})$, $r_{i-1}=\arg\min_{r_{i-2}<r<T} f_{i-1}(r|r_1,\ldots,r_{i-2})$, $r_{i+1}=\arg\min_{r_i<r<T} f_{i+1}(r|r_1,\ldots,r_i)$ *and* $2r_i-r_{i-1}<T$, *then it holds:*

$$r_i-r_{i-1}\le r_{i+1}-r_i, \tag{52}$$

$$r_{i+1}\ge 2r_i-r_{i-1}. \tag{53}$$

We define $r_0=0$, *so we have:*

$$r_2\ge 2r_1. \tag{54}$$

Proof. This theorem can be proved by using Theorem 2 and the properties of $p_i(r|r_{i-1})$ and $p_{i+1}(r|r_i)$

The curves of $p_{i+1}(r|r_i)$ with $r>r_i$ and $p_i(r|r_{i-1})$ with $r>r_{i-1}$ have the similar shapes according to Fig. 2. The difference between $p_i(r|r_{i-1})$ and $p_{i+1}(r|r_i)$ can be characterized by their saturation points $r_i^*=\arg\min_r\{r|1-p(r|r_{i-1})\le\varepsilon\}$ and $r_{i+1}^*=\arg\min_r\{r|1-p(r|r_i)\le\varepsilon\}$. Define increasing step $\Delta r_i^*=r_i^*-r_{i-1}$ and $\Delta r_{i+1}^*=r_{i+1}^*-r_i$, then $r_i^*=r_{i-1}+\Delta r_i^*$ and $r_{i+1}^*=r_i+\Delta r_{i+1}^*$ can be rewritten as:

$$\Delta r_i^*=\arg\min_{\Delta r}\{\Delta r|1-p(r_{i-1}+\Delta r|r_{i-1})\le\varepsilon\}, \tag{55}$$

$$\Delta r_{i+1}^*=\arg\min_{\Delta r}\{\Delta r|1-p(r_i+\Delta r|r_i)\le\varepsilon\}. \tag{56}$$

Now investigate the curves of $p(r_{i-1}+\Delta r|r_{i-1})$ and $p(r_i+\Delta r|r_i)$ (see Fig. 6). According to the property (i.e., (42)) of $p(r_i|r_{i-1})$, $\Delta r_{i+1}^*>\Delta r_i^*$ holds because $r_i>r_{i-1}$. □

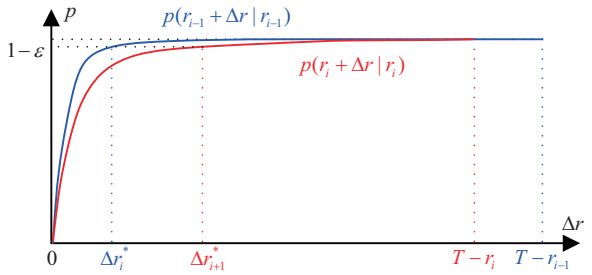


Fig. 6. $p(r_{i-1}+\Delta r|r_{i-1})$ and $p(r_i+\Delta r|r_i)$.

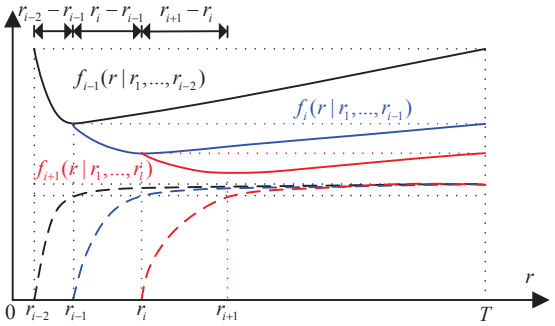


Fig. 7. Illustration of Theorem 5.

Fig. 7 illustrates the nature of Theorem 5 where the objective functions and estimated optimal solution at saturation points are shown. The relationship of r_{i-1}, r_i, and r_{i+1} are $r_{i-1}-r_{i-2}<r_i-r_{i-1}<r_{i+1}-r_i$ (i.e., $\Delta r_{i-1}<\Delta r_i<\Delta r_{i+1}$).

According to Theorem 5, if $r_i=\arg\min_{r_{i-1}<r<T} f_i(r|r_1,\ldots,r_{i-1})$ and $r_{i-1}=\arg\min_{r_{i-2}<r<T} f_i(r|r_1,\ldots,r_{i-2})$ are already known, then the search range for r_{i+1} will be reduced to $[r_i+(r_i-r_{i-1}),T]$ (i.e., $[2r_i-r_{i-1},T]$) where r_i-r_{i-1} is called increasing step.

The training process is shown in Fig. 8 where the increasing phenomenon is used for efficient minimization.

V. Proposed Method: Joint-Minimum Based Multi-Stage Cascade

A. Existence and Uniqueness of a Jointly Optimal Solution

The method in Section IV is a greedy optimization algorithm because it seeks an optimal r_i on the condition that $(r_1,\ldots,r_{i-1})$ are known and fixed. The objective function is $f_i(r|r_1,\ldots,r_{i-1})$, $i=1,\ldots,S$. In this section, we give an algorithm for jointly seeking the optimal $(r_1,\ldots,r_S)$ that globally minimizes the objective function $f(r_1,\ldots,r_S)$ instead

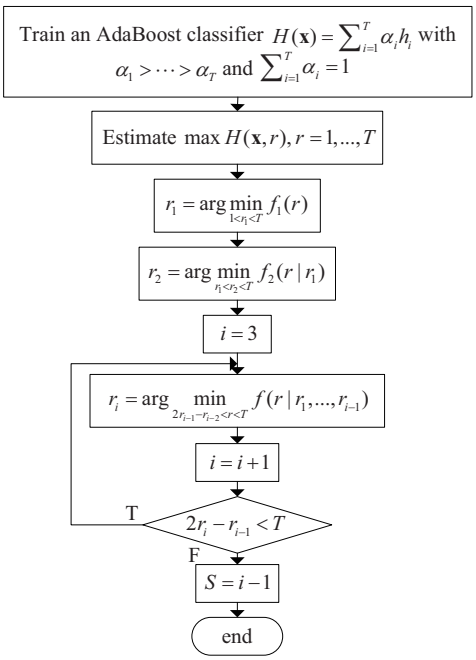


Fig. 8. Local-minimum based multi-stage cascade learning.

of $f_S(r|r_1,...,r_{S-1})$. That is, the goal of joint optimization is to find $(r_1^*,...,r_S^*)=\arg\min_{r_1,...,r_S} f_S(r_1,...,r_S)$

For the sake of clarity, we start with establishing a globally optimal two-stage cascade structure. The globally optimal cascade structure with more than two stages will be extended from the two-stage one.

The goal of jointly optimal two-stage cascade learning aims at finding $(r_1^*,r_2^*)=\arg\min_{r_1,r_2} f_2(r_1,r_2)$.

Obviously, if both $f_2(r|r_1)=f_2(r_1,r)$ and $f_2(r|r_2)=f_2(r,r_2)$ have unique minimization solutions, then $f_2(r_1,r_2)$ has unique minimization solutions. $f_2(r|r_1)$ means the objective function of a two-stage cascade where the parameter r_1 of stage 1 is known and the parameter r of stage 2 is a unknown variable. $f_2(r|r_2)$ stands for the situation where the parameter r_2 of stage 2 is known and the parameter r of stage 1 is a variable. The theorems related to the jointly optimization are as follows.

Theorem 6. $\min_r f_2(r,r_2)=\min_r f_2(r|r_2)$ *has a unique minimum solution* r_1.

Proof. We first prove the existence of the minimum solution and then give the evidence of the uniqueness of the minimum solution.

Existence:

$$\begin{aligned}\because f_2(r|r_2) &= p_1(r)(r+c)+(1-p_1(r))p_2(r_2|r)(r_2+2c)\\ &\quad +(1-p_1(r))(1-p_2(r_2|r))(T+3c),\\ \therefore f_2'(r|r_2) &= p_1'(r)(r-2c-T)+p_1(r)\\ &\quad +(T+c-r_2)p_1'(r)p_2(r_2|r)\\ &\quad -(T+c-r_2)(1-p_1(r))p_2'(r_2|r).\end{aligned}$$

Because the sum of rejected negative sub-windows of stage 1 and stage 2 is a const $\eta>0$ once r_2 is fixed:

$$p_1(r)+(1-p_1(r))p_2(r_2|r)=p_1(r_2)=\eta. \tag{57}$$

Computing the derivative of r to both sides of (57) yields:

$$p_1'(r)p_2(r_2|r)-(1-p_1(r))p_2'(r_2|r)=p_1'(r). \tag{58}$$

Therefore, we can get $f_2'(r|r_2)$ as

$$f_2'(r|r_2)=p_1'(r)(r-r_2-c)+p_1(r). \tag{59}$$

$\because \lim_{r\to 0} p_1(r)=0$ (see (19)) and $p_1'(r)\geq 0$ (see (17)),
$\therefore \lim_{r\to 0} f_2'(r|r_2)=-(r_2+c)p_1'(0)<0$.
$\because \lim_{r\to r_2+c} f_2'(r|r_2)=p_1(r_2+c)>0$.
$\because \lim_{r\to r_2+c} f_2'(r|r_2)>0$, $\lim_{r\to 0} f_2'(r|r_2)<0$ and $f_2'(r|r_2)$ is a continuous function,
$\therefore$ It must exist a $r_1\in[1,r_2+c)$ satisfying $f_2'(r_1|r_2)=0$ and $r_1=\arg\min f_2(r|r_2)$.

Uniqueness (Proof by contradiction):

Suppose there are two local minimum solutions $\tilde{r}_1$ and r_1 with $\tilde{r}_1<r_1$. Then it holds that $f_2'(\tilde{r}_1|r_2)-f_2'(r_1|r_2)=0$.

Now investigate the value of $f_2'(\tilde{r}_1|r_2)-f_2'(r_1|r_2)$ = $[p_1'(\tilde{r}_1)(\tilde{r}_1-r_2-c)-p_1'(r_1)(r_1-r_2-c)]+[p_1(\tilde{r}_1)-p_1(r_1)]$ if $\tilde{r}_1<r_1$ is true.

$\because \tilde{r}_1<r_1$,
$\therefore p'(\tilde{r}_1)>p'(r_1)>0, \tilde{r}_1-r_2-c<r_1-r_2-c<0, 0<p(\tilde{r}_1)<p(r_1)$,
$\therefore f_2'(\tilde{r}_1|r_2)-f_2'(r_1|r_2)<0$.

This contradicts $f_2'(\tilde{r}_1|r_2)-f_2'(r_1|r_2)=0$. Therefore, $\tilde{r}_1<r_1$ is wrong.

Similarly, we can prove that $\tilde{r}_1>r_1$ is wrong. Consequently, $\tilde{r}_1=r_1$ is true which means a unique solution in $r_1\in[1,r_2+c)$. Because c is smaller than 1 (see the statement below (13)), It is equivalent that the unique solution r_1 is in the range $[1,r_2)$ □

Theorem 6 tells that if the information of stage 2 is given, then one can find an optimal parameter r for stage 1 so that the computation cost f_2 of the final two-stage cascade is minimized.

Theorem 7. $f_2(r_1,r_2)$ *has a unique minimum solution* (r_1^*,r_2^*).

Proof. Because both $\min_r f_2(r|r_1)=\min_r f_2(r_1,r)$ (see Theorem 4) and $\min_r f_2(r|r_2)=f_2(r,r_2)$ (see Theorem 6) have unique minimum solutions, so $\min f_2(r_1,r_2)$ has a unique minimum solution. That is $f_2(r_1^*,r_2^*)=\min_{r_1,r_2} f_2(r_1,r_2)=\min_{r_1}\min_{r_2} f_2(r_2|r_1)=\min_{r_2}\min_{r_1} f_2(r_1|r_2)$ □

It is straightforward to generalize Theorem 7 to the following Theorem:

Theorem 8. $f_i(r_1,...,r_i)$ *has a unique minimum solution* $(r_1^*,...,r_i^*)$.

B. How to Search the Jointly Optimal Solutions

1) Algorithm: Theorem 8 guarantees the existence and uniqueness of jointly optimizing the stages of a cascade. In this section, we give algorithms (i.e., Algorithms 2 and 3) for searching the solution and then theoretically justify the algorithms in theory. We start with the algorithm for

Algorithm 2 Globally optimal two-stage cascade learning.

Input:

Strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$ and its threshold t;

A set of true negative sub-windows$\{\mathbf{x}|l(\mathbf{x}) = -1\}$;

Output:

$(r_1^*, r_2^*) = \arg\min_{r_1,r_2} = f_2(r_1, r_2)$;

1: **Initialization**

2: Search the optimal solution r_1 of $f_1(r)$ for stage 1 in the range of $(1, T)$: $r_1 = \arg\min_{1<r<T} f_1(r)$.

3: Given r_1, search the optimal solution r_2 of $f_2(r|r_1)$ for stage 2 in the range of $[2r_1, T)$: $r_2 = \arg\min_{2r_1 \le r < T} f_2(r|r_1)$. See Theorem 5 for the reason of $r \ge 2r_1$.

4: **Iteration**

5: Given r_2, search the optimal solution $\tilde{r}_1$ of $f_2(r|r_2)$ in the range of $1 < r \le r_1$ for stage 1: $\tilde{r}_1 = \arg\min_{1<r\le r_1} f_2(r|r_2)$. Note that $\tilde{r}_1 < r_1$ (see Theorem 9). An efficient search strategy is decreasing r from r_1 step by step until $f_2(r|r_2)$ does not decrease. $f \leftarrow f_2(r_1|r_2)$.

6: Given $\tilde{r}_1$, search the optimal solution $\tilde{r}_2$ of $f_2(r|\tilde{r}_1)$ for stage 2 in the range of $\tilde{r}_1 < r \le r_2$: $\tilde{r}_2 = \arg\min_{\tilde{r}_1<r\le r_2} f_2(r|\tilde{r}_1)$. Note that $\tilde{r}_2 \le r_2$ (see Theorem 12). An efficient search strategy is decreasing r from r_2 step by step until $f_2(r|\tilde{r}_1)$ does not decrease. $\tilde{f} \leftarrow f_2(r|\tilde{r}_1)$.

7: Update $r_1 \leftarrow \tilde{r}_1$, $r_2 \leftarrow \tilde{r}_2$.

8: **until** $f - \tilde{f} \ge \mu$

9: **return** $r_1^* \leftarrow \tilde{r}_1$, $r_2^* \leftarrow \tilde{r}_2$.

optimizing a two-stage cascade and then generalize it to multi-stage one.

The task of jointly optimizing a two-stage cascade can be expressed as $(r_1^*, r_2^*) = \arg\min_{r_1,r_2} f_2(r_1, r_2)$. The idea of our optimization method is shown in Algorithm 2.

The proposed Algorithm 2 is an alternative optimization procedure. In the initialization step, the solution r_1 of the one-stage cascade learning is searched in the largest range $1 < r < T$: $r_1 = \arg\min_{1<r<T} f_1(r)$. The value of r_1 is shown in Fig. 9, where "#1" means that r_1 is obtained firstly. The obtained r_1 is used as the upper bound of the searching range for the better solution $\tilde{r}_1$ in line 5 of Algorithm 2. After r_1 is given, line 3 of Algorithm 2 searches the optimal solution r_2 of $f_2(r|r_1)$ for stage 2 in the range of $2r_1 \le r < T$: $r_2 = \arg\min_{2r_1\le r<T} f_2(r|r_1)$. Based on (54), the search range starts from $2r_1$. The value of r_2 is shown in Fig. 9, where "#2" means that r_2 is the second value obtained by Algorithm 2.

In line 5 of Algorithm 2, r_2 is given and the task is to search the optimal solution $\tilde{r}_1$ of $f_2(r|r_2)$ in the range of $1 < r \le r_1$ for stage 1: $\tilde{r}_1 = \arg\min_{1<r\le r_1} f_2(r|r_2)$. Because $r_1 \ll T$, the search range $1 < r \le r_1$ is much smaller than the one (i.e., $1 < r < T$) in line 2. Theorem 9 guarantees $\tilde{r}_1 \le r_1$ for the first round of iteration. $\tilde{r}_1$ is the third value

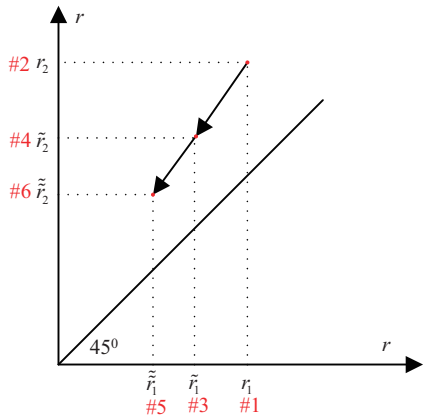


Fig. 9. Illustration of the intermediate values obtained by Algorithm 2. r_1 and r_2 are the outputs of Initialization. The sequence of the stage parameters are updated in the following turn: $r_1 \to r_2 \to \tilde{r}_1 \to \tilde{r}_2 \to \tilde{\tilde{r}}_1 \to \tilde{\tilde{r}}_2$ with $\tilde{\tilde{r}}_1 < \tilde{r}_1 < r_1$ and $\tilde{\tilde{r}}_2 < \tilde{r}_2 < r_2$

obtained by Algorithm 2 which is shown near "#3" in Fig. 9. Experimental results and intuitive analysis show that the absolute distance $|\tilde{r}_1 - r_1|$ from $\tilde{r}_1$ to r_1 is much smaller than the absolute distance $|1 - \tilde{r}_1|$ from 1 to $\tilde{r}_1$, the search strategy of decreasing r from r_1 step by step until $f_2(r|r_2)$ does not decrease is more efficient than the one of increasing r from 1 step by step until $f_2(r|r_2)$ does not increase.

In line 6 of Algorithm 2, $\tilde{r}_1$ is given and the task is to search the optimal solution $\tilde{r}_2$ of $f_2(r|\tilde{r}_1)$ in the range of $\tilde{r}_1 < r \le r_2$ for stage 2: $\tilde{r}_2 = \arg\min_{\tilde{r}_1<r\le r_2} f_2(r|\tilde{r}_1)$. Because $r_2 < T$, the upper bound of the search range is much smaller than the one (i.e., T) in line 3. Moreover, as iteration runs, the updated r_2 becomes smaller and so the upper bound of search range for $\tilde{r}_2$ becomes smaller too. Theorem 10 guarantees $\tilde{r}_2 \le r_2$. The value of $\tilde{r}_2$ is shown in Fig. 9 which is "#4" obtained by Algorithm 2. Experimental results and intuitive analysis show that the absolute distance $|\tilde{r}_2 - r_2|$ from $\tilde{r}_2$ to r_2 is much smaller than the absolute distance $|\tilde{r}_1 - \tilde{r}_2|$ from $\tilde{r}_1$ to $\tilde{r}_2$, the search strategy of decreasing r from r_2 step by step until $f_2(r|\tilde{r}_1)$ does not decrease is more efficient than the one of increasing r from $\tilde{r}_1$ step by step until $f_2(r|\tilde{r}_1)$ does not increase.

In the second round of iteration, because $\tilde{r}_2 \le r_2$, the parameter value $\tilde{\tilde{r}}_1$ for stage 1 is obtained and shown in Fig. 9 with a label "#5". According to Theorem 11, it is true that $\tilde{\tilde{r}}_1 \le \tilde{r}_1$. Subsequently, the parameter value $\tilde{\tilde{r}}_2$ for stage 2 is obtained and shown in Fig. 10 with a label "#6". According to Theorem 10, it is true that $\tilde{\tilde{r}}_2 \le \tilde{r}_2$.

The iteration stops if the difference between the value f of objective function in line 5 of Algorithm 2 and the one $\tilde{f}$ in in line 6 of Algorithm 2 is equal to or smaller than the threshold $\mu \ge 0$.

Decreasing Phenomenon: Fig. 9 shows an interesting phenomenon: (1) Once a new stage 2 is added, the parameter r_1 of stage 1 should be updated by decreasing r_1 to a smaller number $\tilde{r}_1$ so that the computation cost is minimized. (2) Once the number of stages is fixed, the parameter for each stage decreases gradually as iteration proceeds.

2) Justification of the Algorithm: Theorems 9-12 are to given to theoretically interpret the so-called Decreasing Phenomenon and justify Algorithm 2. Theorem 9 implies that the parameter r_1 of stage 1 should be updated by decreasing to a small number when the parameter r_2 of stage 2 is fixed.

Theorem 9. *$\tilde{r}_1 = \arg\min_r f_2(r|r_2) \leq \arg\min_r f_1(r) = r_1$ where $r_2 > r_1$.*

Proof.
$\because \tilde{r}_1 = \arg\min_r f_2(r|r_2)$ and $r_1 = \arg\min_r f_1(r)$,
$\therefore \left.\frac{df_2(r|r_2)}{dr}\right|_{r=\tilde{r}_1} - \left.\frac{df_1(r)}{dr}\right|_{r=r_1} = 0.$
$\because f_1(r) = p_1(r)(r+c) + (1-p_1(r))(T+2c)$,
$\therefore f_1'(r_1) = p_1(r_1) + p_1'(r_1)(r_1 - T - c)$.
$\because f_2(r|r_2) = p_1(r)(r+c) + (1-p_1(r))[p_2(r_2|r)(r_2+2c) + (1-p_2(r_2|r))(T+3c)]$,
$\therefore f_2'(r|r_2) = p_1'(r)(r - r_2 - c) + p_1(r)$ (see (59)).
Now investigate the value of $f_2'(\tilde{r}_1|r_2) - f_1'(r_1) = p_1'(\tilde{r}_1)(\tilde{r}_1 - r_2 - c) - p_1'(r_1)(r_1 - T - c) + (p_1(\tilde{r}_1) - p_1(r_1))$ if $\tilde{r}_1 > r_1$ is true:
$\because \tilde{r}_1 > r_1$ is assumed,
$\therefore 0 < p_1'(\tilde{r}_1) < p_1'(r_1), 0 > \tilde{r}_1 - r_2 - c > r_1 - T - c, p_1(\tilde{r}_1) > p_1(r_1) > 0$,
$\therefore p_1'(\tilde{r}_1)(\tilde{r}_1 - r_2 - c) > p_1'(r_1)(r_1 - T - c), p_1(\tilde{r}_1) > p_1(r_1)$,
$\therefore f_2'(\tilde{r}_1|r_2) - f_1'(r_1) > 0$.
This contradicts $f_2'(\tilde{r}_1|r_2) - f_1'(r_1) = 0$. So $\tilde{r}_1 > r_1$ is wrong and $\tilde{r}_1 \leq r_1$ is true. □

As a lemma of Theorem 9, we have the following theorem:

Theorem 10. *If $(r_1^{i*}, ..., r_i^{i*}) = \arg\min_{r_1,...,r_i} f_i(r_1, ..., r_i)$ and $(r_1^{*(i+1)}, ..., r_i^{*(i+1)}, r_{i+1}^{*(i+1)}) = \arg\min_{r_1,...,r_i,r_{i+1}} f_i(r_1, ..., r_i, r_{i+1})$, then $r_j^{*(i+1)} \leq r_j^{*i}$, $j = 1, ..., i$.*

As a generalized version of Theorem 9, Theorem 10 tells that once a new stage $i+1$ is added, all the optimal parameters of the existing stages $1, \ldots, i$ should be updated and decreased so that the computation cost is minimized.

Theorem 11. *If $\tilde{r}_2 < r_2$, then $\tilde{r}_1 = \arg\min_r f_2(r|\tilde{r}_2) \leq \arg\min_r f_2(r|r_2) = r_1$.*

Proof.
$\because \tilde{r}_1 = \arg\min_r f_2(r|\tilde{r}_2)$ and $r_1 = \arg\min_r f_2(r|r_2)$,
$\therefore \left.\frac{df_2(r|\tilde{r}_2)}{dr}\right|_{r=\tilde{r}_1} - \left.\frac{df_2(r|r_2)}{dr}\right|_{r=r_1} = 0$ is true.
Now investigate the value of $f_2'(\tilde{r}_1|\tilde{r}_2) - f_2'(r_1|r_2) = [p_1'(\tilde{r}_1)(\tilde{r}_1 - \tilde{r}_2 - c) - p_1'(r_1)(r_1 - r_2 - c)] + [p_1(\tilde{r}_1) - p_1(r_1)]$ if $\tilde{r}_1 > r_1$ is true:
$\because \tilde{r}_1 > r_1$ is assumed,
$\therefore 0 < p_1'(\tilde{r}_1) < p_1'(r_1), 0 > \tilde{r}_1 - \tilde{r}_2 - c > r_1 - r_2 - c, p_1(\tilde{r}_1) > p_1(r_1) > 0$,
$\therefore p_1'(\tilde{r}_1)(\tilde{r}_1 - \tilde{r}_2 - c) > p_1'(r_1)(r_1 - r_2 - c), p_1(\tilde{r}_1) > p_1(r_1)$,
$\therefore f_2'(\tilde{r}_1|\tilde{r}_2) - f_2'(r_1|r_2) > 0$.
This contradicts $f_2'(\tilde{r}_1|\tilde{r}_2) - f_2'(r_1|r_2) = 0$. So $\tilde{r}_1 > r_1$ is wrong and $\tilde{r}_1 \leq r_1$ is true. □

Theorem 12. *If $\tilde{r}_1 < r_1$, then $\tilde{r}_2 = \arg\min_{\tilde{r}_1 < r < T} f_2(r|\tilde{r}_1) \leq \arg\min_{r_1 < r < T} f_2(r|r_1) = r_2$.*

Algorithm 3 Globally optimal multi-stage cascade learning.

Input:
Strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$ and its threshold t;
A set of true negative sub-windows$\{\mathbf{x}|l(\mathbf{x}) = -1\}$;

Output:
$(r_1^*, ..., r_S^*) = \arg\min_{r_1,...,r_S} = f_S(r_1, ..., r_S)$ where S is the number of stages in the final cascade structure;

1: Search the optimal solution $r_1^{*(1)}$ of $f_1(r)$ for stage 1 in the range of $1 < r < T$: $r_1 = \arg\min_{1<r<T} f_1(r)$. $f \leftarrow f_1(r_1)$;
2: **for** $i = 2$ to S **do**
3: Initialize the upper bound $r_1^u, ..., r_{i-1}^u$ of $r_1, ..., r_{i-1}$: $r_j^u \leftarrow r_j^{*(i-1)}$ for $j = 1, ..., i-1$;
4: Initialize the upper bound r_i^u of r_i by finding $r_i^u = \arg\min_{r_i \geq 2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)},} f_i\left(r_i | r_1^{*(i-1)}, ..., r_{i-1}^{*(i-1)}\right)$ with the search range $r_i \geq 2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)}$ and $r_0^{*(1)} \triangleq 0$.
5: $f \leftarrow f_i(r_1^{*(i-1)}, ..., r_{i-1}^{*(i-1)})$, $\tilde{f} \leftarrow f_i(r_1^u, ..., r_i^u)$.
6: **while** $f - \tilde{f} > \varepsilon$ **do**
7: $f \leftarrow \tilde{f}$;
8: **for** $j = 0$ to i **do**
9: $r_j^* = \arg\min_{r_i \leq r_j^u} f_i(r_j | r_k^u, k \neq j)$;
10: **end for**
11: $\tilde{f} \leftarrow f_i(r_1^*, ..., r_i^*)$, $r_j^u \leftarrow r_j^*$.
12: **end while**
13: $r_j^{*i} \leftarrow r_j^u$, $j = 1, ..., i$;
14: **end for**
15: **return** $r_i^* \leftarrow r_i^{*S}$, $i = 1, ..., S$.

Proof. See Appendix A. □

Theorems 9, 11 and 12 can be extended to multi-stage cascade. Correspondingly, Decreasing Phenomenon can be generalized to Generalized Decreasing Phenomenon and Algorithm 2 can be generalized to Algorithm 3.

Generalized Decreasing Phenomenon: If the alternative optimization algorithm 3 is used to find the globally optimal solution $(r_1^{*i}, r_2^{*i}, ..., r_i^{*i}) = \arg\min_{r_1,...,r_i,} f_i(r_1, ..., r_i)$, then it holds that:

(1) Once a new stage $i+1$ is added, all the optimal parameters of the existing stages $1, \ldots, i$ are updated and decreased so that the computation cost is minimized.

(2) Once the number of stages is fixed, the parameter for each stage decreases gradually as iteration proceeds.

In Algorithm 3, $\tilde{f}$ is the objective function after a new stage i is added while f is the one before stage i is added. That is , f is the value of objective function when there are $i-1$ stages. According to Theorem 5, when a new stage i is to be added, the optimal solution r_i can be searched by increasing r_i from $2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)}$ instead of $r_{i-1}^{*(i-1)}$. Because $2r_{i-1}^{*(i-1)} - r_{i-2}^{*(i-1)}$ is much larger than $r_{i-1}^{*(i-1)}$, the search efficiency is very high. The iteration in line 5 of Algorithm 3 stops if the difference between f and $\tilde{f}$ is below a threshold $\varepsilon > 0$, which implies that the algorithm arrives at global minimum solution for i stages.

Fig. 10 shows the classification procedure of multi-stage

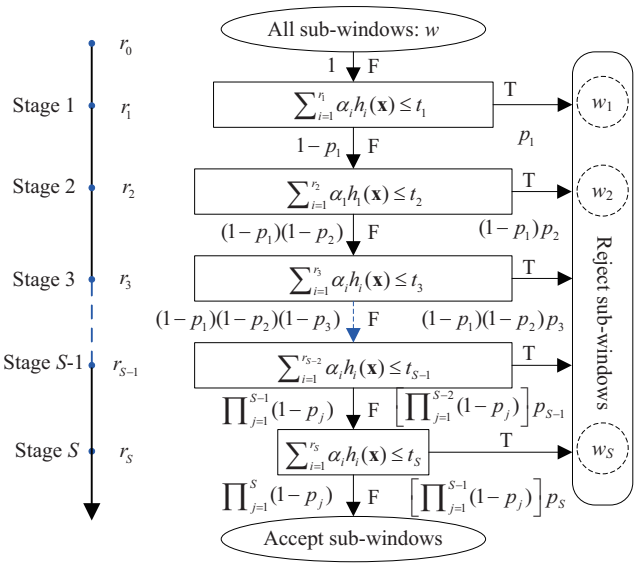


Fig. 10. The classification procedure of the multi-stage iCascade algorithm.

iCascade where the partition points (r_1,...,r_S) are given by Algorithm 3. If the computation cost of classifying positive samples is neglected, the computation cost f_S of iCascade can be estimated by

$$f_S = \sum_{i=1}^{S} (r_i + ic) \left[\prod_{j=1}^{i} (1 - p_{j-1}(r_{j-1})) \right] p_i(r_i) + (T + (S+1)c) \left[\prod_{j=1}^{S+1} (1 - p_{j-1}(r_{j-1})) \right]. \quad (60)$$

C. Threshold learning in iCascade

Once the number of weak classifiers in each stage is determined by Algorithm 3, the parameters affecting the computation cost are the thresholds t_i, $i = 1, \ldots, S$. In this section, we give theorem and algorithm for setting the thresholds (t_1,..., t_S).

Theorem 13 tells that the computation cost f_S monotonically decreases with t_i and $p_i(t_i)$, $i = 1, ..., S$. So the computation cost can be reduced by decreasing the thresholds under the constraint of minimum-acceptable detection rate.

Theorem 13. *f_S monotonically decreases with t_i and $p_i(t_i)$, $i = 1, ..., S$.*

Proof. See Appendix B. □

If the detection rate $D = 1$ (i.e., all the positive training samples are correctly classified) is the constraint, then the optimal threshold t_i^* can be expressed as:

$$t_i^* = \arg\min t_i, \quad \text{s.t. } d(t_i) = 1, \quad i = 1, ..., S. \quad (61)$$

which guarantees $D = \prod_{i=1}^{S} d(t_i^*) = 1$. In (61), $d(t_i)$ is the detection rate of stage i defined by:

$$d(t_i) = \frac{\sum_{\mathbf{x}} I(\sum_{j=1}^{r_i} \alpha_j h_j(\mathbf{x}) > t)}{\sum_{\mathbf{x}} I(l(\mathbf{x}) == 1)}. \quad (62)$$

It is challenging to choose the optimal thresholds if the expected detection $D < 1$. It is well known that the detection rate D of the system is the product of the detection rate $d(t_i)$ of each stage. A popular way to set $d(t_i)$ is

$$d(t_i) = D^{1/S}, \ i = 1, ..., S. \quad (63)$$

However, when the number of stages of iCascade is very large, it holds that $d(t_i) \approx 1$. Such high $d(t_i)$ makes the threshold t_i very large and the corresponding computation cost is very large.

To deal with the above problem, we propose to use Algorithm 4 for threshold learning. The initial thresholds are chosen by (63) guaranteeing the detection rate D being 1. The corresponding initial computation cost is denoted by f_S. The main issue is to select which stage to decrease its initial threshold by a small step Δt_i. In our algorithm, the derivative f'_S of the computation cost f_S against detection rate D is computed by

$$f'_S(i) \approx \Delta f_S / \Delta D_i, \quad (64)$$

where ΔD_i is the variation of the system detection rate. Note that the variation ΔD_i is caused by changing t_i to $t_i - \Delta t_i$ while the thresholds t_k of other stages (i.e., $k \neq i$) remain unchanged.

The stage j with the largest derivative is selected and its threshold t_j is then decreased by the small step Δt_j:

$$j = \arg\max_i \frac{\Delta f_S}{\Delta D_i}, \quad (65)$$

$$t_j \leftarrow t_j - \Delta t_j, \quad (66)$$

with the thresholds of the stages (i.e., $i \neq j$) unchanged.

Re-compute the computation cost f_S and detection rate D when t_j is updated:

$$f_S \leftarrow f_S - \Delta f_S, \quad (67)$$

$$D \leftarrow D - \Delta D_j. \quad (68)$$

The step Δt_i is small enough to keep the detection rate D smaller than the target detection rate D_o.

As shown in Algorithm 4, the iteration of choosing the most important stage $j = \arg\max_i \Delta f_S / \Delta D_i$, updating its threshold $t_j \leftarrow t_j - \Delta t_j$ and corresponding computation cost $f_S \leftarrow f_S - \Delta f_S$ and detection rate $D \leftarrow D - \Delta D_j$ runs until the updated detection rate D is below the expected detection rate D_o.

VI. EXPERIMENTAL RESULTS

A. Experimental Setup

The classical cascade learning algorithms of Viola and Jones (VJ)(a.k.a., Fixed Cascade) [8], Recyling Cascade [10] and Recyling & Retracting Cascade [10] are compared with the proposed iCascade algorithm. The testing dataset is the standard MIT-CMU frontal face databaset [33], [8]. The positive training dataset consists of about 20000 normalized face

Algorithm 4 Threshold learning algorithm for iCascade.

Input:

Expected detection rate D_o;
Positve and negative training samples;
Strong classifiers $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$;

Output:

The optimal thresholds t_i of all the S stages;

1: Initialize the thresholds t_i for each stage by $t_i = \arg\min t_i$, s.t. $d(t_i) = 1$, $i = 1, ..., S$ so that the system detection rate $D = 1$;
2: Corresponding to the initial thresholds, the initial computation cost of the system is computed by (60) and denoted by f_S;
3: **repeat**
4: For each stage, compute the approximation of the derivative $\Delta f_S / \Delta D_i$ of the computation cost f_S against detection rate D. The variations Δf_S and ΔD_i are caused by changing t_i to $t_i - \Delta t_i$ while the thresholds t_k of other stages (i.e., $k \neq i$) remain unchanged;
5: From all the S stages, choose the stage j with largest derivative $j = \arg\max_i \Delta f_S / \Delta D_i$. Then decrease the threshold t_j of the stage j by a small step Δt_j : $t_j \leftarrow t_j - \Delta t_j$;
6: Update the computation cost f_S and detection rate D: $f_S \leftarrow f_S - \Delta f_S$, $D \leftarrow D - \Delta D_j$;
7: **until** $D \leq D_o$
8: **return** the updated thresholds t_i of all the S stages.

images of size 20×20 pixels. 5000 non-face large images are collected from web sites to generate negative training dataset. Both of the positive and negative training images can be downloaded from https://sites.google.com/site/yanweipang/publica.

In addition, the intermediate results demonstrating the correctness of the proposed theorems are given in Section VI.B.

A strong classifier $H(\mathbf{x}) = \sum_{i=1}^{T} \alpha_i h_i(\mathbf{x})$ is considered input of iCascade. The strong classifier is obtained by standard AdaBoost algorithm without designing of cascade structure.

B. Intermediate Results of iCascade

Some intermediate results are shown in this section. These results show the rationality of the assumptions and the correctness of the proposed theorems.

1) Local-Minimum Based Cascade: In Section III, the regular strong AdaBoost classifier is divided into $H_L(\mathbf{x}, r)$ and $H_R(\mathbf{x}, r)$ to reject some negative sub-windows earlier, and the key problem is to determine an optimal r to minimize the computation cost. To solve this problem, it is necessary to reveal the relationship between r and the negative rejection rate p.

In this part, with the training dataset described in Section VI.A, we train a regular strong AdaBoost classifier and split it into two parts by r, which varies 1 to T. In the case that detection rate is fixed at 1, Fig. 11 shows that the negative rejection rate p increases with r. p first grows quickly from 0 to 0.96 when r changes from 1 to a small value $r^* = 80$, and then becomes stable when r is larger than r^* . Thus, we can model

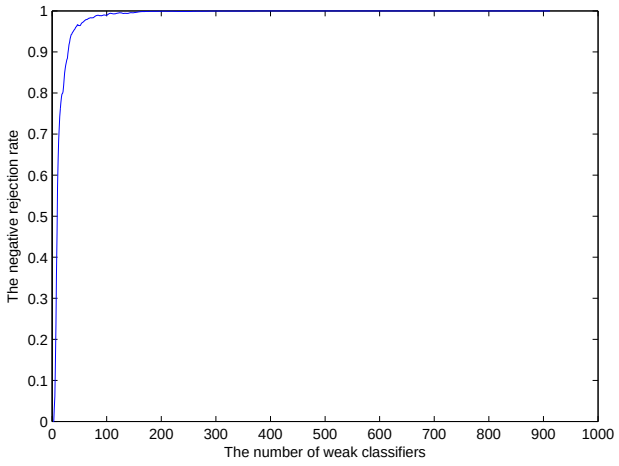


Fig. 11. The negative rejection rate $p(r)$

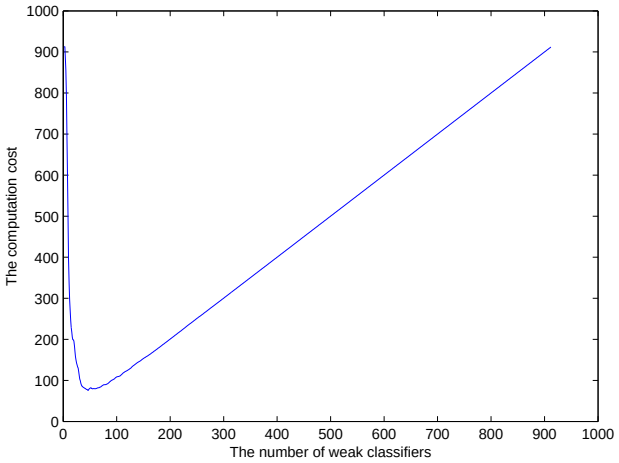


Fig. 12. The computation cost $f(r)$

$p(r)$ by combining of two linear functions: $p_1(r) = 0.012r$ with $r < r^*$ and $p_2(r) = 1$ with $r \geq r^*$. Fig. 11 demonstrates the rationality of (16)-(25). Fig. 12 shows that the computation cost f first decreases and then increases with r, and the unique minimum is nearby r^*. Fig. 12 experimentally proves the correctness of Theorem 1 and Theorem 2.

When we split the regular strong AdaBoost classifier into $H_L(\mathbf{x}, r_1)$ and $H_R(\mathbf{x}, r_1)$, the sub-windows not rejected by stage 1 are fed to stage 2. Then we can divide $H_R(\mathbf{x}, r_1)$ into two parts to form a 2-stage cascade. In this process, we should know some properties of the negative rejection rate of stage 2 (i.e., $p(r|r_1)$). Fig. 13 shows how $p(r|r_1)$ changes with r, where the curves of $p(r|r_1)$ when $r_1 = 10$ and $r_1 = 30$ are given, respectively. $p(r|r_1)$ has the similar characteristics with $p(r)$. Fig. 14 shows how the derivative curves of $p(r|r_1)$ change with r. Obviously, when $\widetilde{r_1} < r_1$, $p(r_2|\widetilde{r}_1) > p(r_2|r_1)$ and $p'(r_2|\widetilde{r}_1) < p'(r_2|r_1)$. Fig. 13 and 14 directly support the correctness of (39)-(43).

We use the local-minimum based multi-stage cascade learning method (see Fig. 4) to train an 8-stage cascade classifier. Table 1 shows how the computation cost f changes with the number of stages. The computation cost first decreases quickly

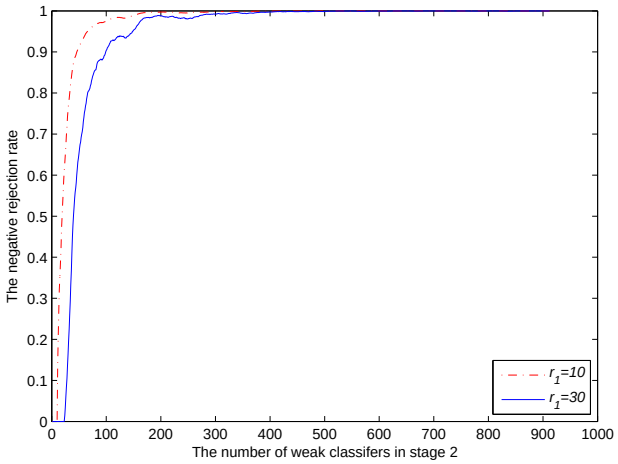


Fig. 13. Some properties of $p(r_2|r_1)$

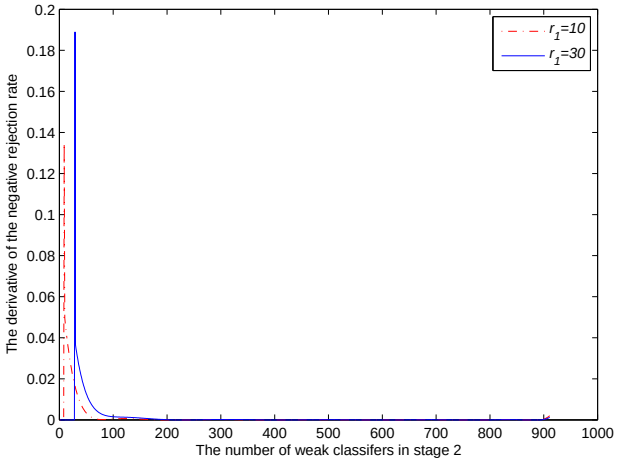


Fig. 14. The derivative of $p(r_2|r_1)$

and then becomes stable. This phenomenon can be understood easily, because the first few stages can reject the most part of the sub-windows, and then only some small part of the sub-windows arrive at last few stages which don't produce much computation cost.

2) Joint-Minimum Based Cascade: In the local-minimum based multi-stage cascade(i.e., Fig. 4), it seeks an optimal r_i on the condition that $(r_1, ..., r_{i-1})$ are known and fixed, so $(r_1, ..., r_{i-1}, r_i)$ can't be jointly optimal for minimizing the computation cost $f(r_1, ..., r_i)$ where not only r_i but also $r_1, \ldots, r_{i-1}$ are variable. Thus, Algorithm 3 is proposed to train the joint-minimum based multi-stage cascade.

Fig. 15 shows the iteration process of Algorithm 3. The number 48 on the top blue line is $r_1^{*(1)} = \arg\min f_1(r)$, which is the result of line 1 of Algorithm 3. The right number 172 on the top red line is $r_2^u = \arg\min f_2(r_2|r_1^{*(1)}) = 172$ (see line 3 of Algorithm 3). Obviously $r_2^u = 172$ is the solution of local-minimum based optimization. The number 17 and 82 on the second blue line are solutions of joint-minimum based optimization (i.e., line 13 of Algorithm 3). Generally, the right most number on each red line is the upper bound r_i^u of Algorithm 3, and the number on each blue

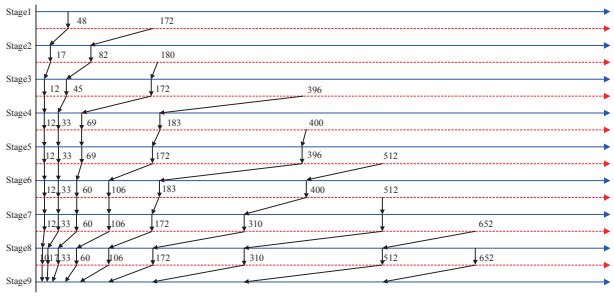


Fig. 15. Generalized decreasing phenomenon in joint-minimum based multi-stage cascade

line are the solutions of joint-minimum based optimization $r_j^{*i}, j = 1, \ldots, i$. The generalized decreasing phenomenon can be found from Fig. 15. For example, r_1 decreases from 48 to 17, 12; r_2 decreases from 172 to 82, 45, 33; r_3 decreases from 180 to 172, 69, 60. Table 2 gives the computation cost of the cascade corresponding to Fig. 15. In Table 2, $f_{local}(1) = 74.42$ is the computation cost $f_1(r_1)$ with $r_1 = 48$, and $f_{local}(2) = 52.87$ is equal to $f_2(r_2|r_1)$ with $r_1 = 48$ and local optimization solution is $r_2 = 172$. Generally, $f_{local}(i)$ means the computation cost $f(r_i|r_1, \ldots, r_{i-1})$. In Table 2, $f_{joint}(i)$ is the computation cost $f(r_1, \ldots, r_i)$ of the proposed joint-minimum algorithm where $r_1, \ldots, r_i$ are all unknown and i is the total number of the stages of the cascade. Note that $f_{joint}(1) = f_{local}(1)$, because there is only one stage in the cascade. However, $f_{joint}(2) = 37.83$ and $f_{joint}(3) = 26.67$ are much less than $f_{local}(2) = 52.87$ and $f_{local}(3) = 31.24$, respectively.

To compare the joint-minimum Algorithm 3 with the local-minimum algorithm (see Fig. 4), we visualize f_{joint} in Table II and f in Table I in Fig. 16. With the number of stages increasing, the computation costs decrease. But the difference is that the joint-minimum based algorithm decreases more quickly than the local-minimum algorithm. For example, when the numbers of stages are 3 and 8, the computation costs of the joint-minimum and local-minimum algorithms are (26.67 and 18.99) and (52.16 and 52.14), respectively. In summary, Fig. 16 demonstrates the advantage and importance of the proposed joint-minimum optimization algorithm.

3) Threshold learning: The thresholds $t_i, i = 1, \ldots, T$ affect the computation cost of iCascade. Algorithm 4 gives the iteration process to choose the threshold of each stage for iCascade. Note that the variation ΔD_i of detection rate is obtained by changing t_i to $t_i - \Delta t_i$. As Δt_i gradually decreases, the detection accuracy increases whereas the training time drastically grows. A set of t_i is evaluated. We find that the performance is stably good if $t_i \leq 0.02$. As a tradeoff, $t_i = 0.01$ is empirically employed. Fig. 17 shows how the computation cost updates in the iteration process of the first 20 stages' thresholds. It can be seen that the computation cost significantly decreases with the iteration. In addition, Fig. 17 shows the convergence of the proposed threshold learning algorithm. Fig. 17 supports the correctness of Theorem 13.

TABLE I
COMPUTATION COST f OF THE ALGORITHM IN FIG. 4 VARIES WITH THE NUMBER s OF STAGES

s	1	2	3	4	5	6	7	8
f	74.42	52.87	52.16	52.15	52.14	52.14	52.14	52.14

TABLE II
COMPUTATION COST OF THE CASCADE TRAINED BY ALGORITHM 3

stage number i	1	2	3	4	5	6	7	8	9
$f_{local}(i)$	74.42	52.87	31.24	26.00	22.16	21.99	21.26	21.19	18.98
$f_{joint}(i)$	74.42	37.83	26.67	22.70	22.06	21.31	21.20	18.99	18.38

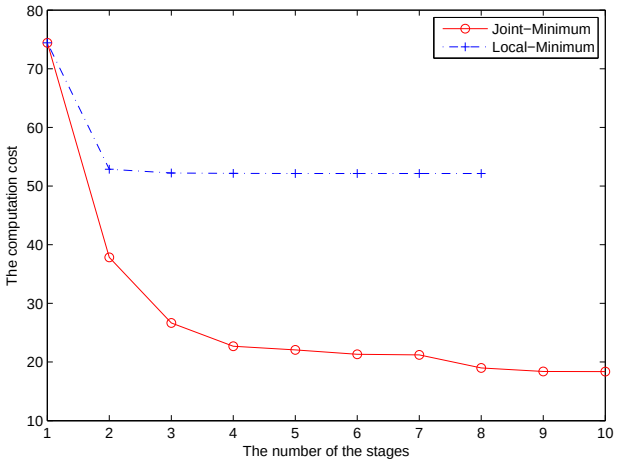


Fig. 16. Comparison of the computation cost between local-minimum based multi-stage cascade and joint-minimum based one

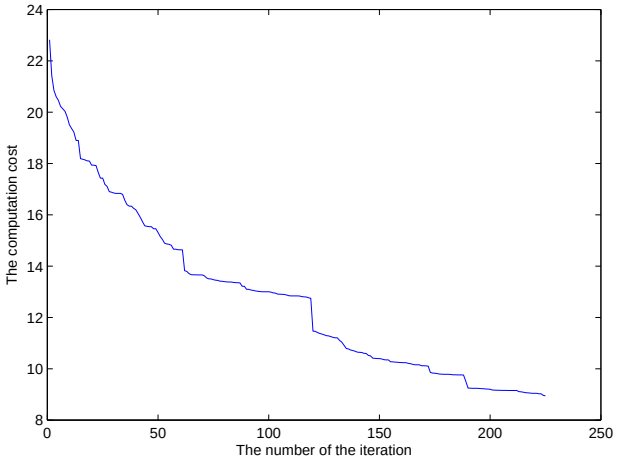


Fig. 17. The computation cost decreases with updation of threshold t_i

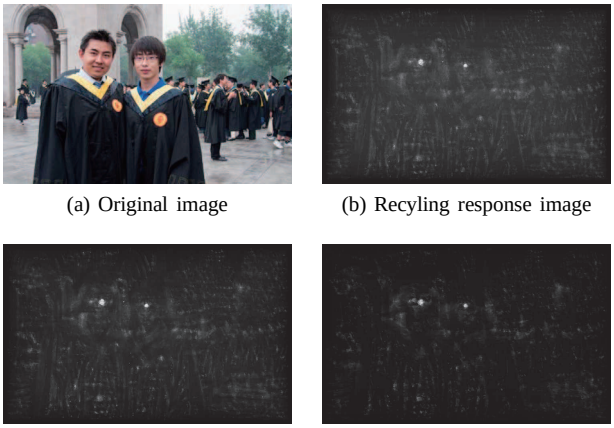
C. Comparison With Other Algorithms

In this section, we compare iCascade with some other algorithms, including Fixed Cascade [8], Recyling Cascade [10] and Recyling & Retracting Cascade [10].

Fixed Cascade is proposed by Viola and Jones. "Fixed" means that the detection rate d_i and the false detection rate f_i of each stage is same and fixed, respectively. If the target detection rate of the cascade is D, the target rejection rate is F and the number of the stages is N, then $d_i = D^{1/N}$ and $f_i = F^{1/N}$. In Recyling Cascade, the score from the previous strong classifier stages serves as a starting point for the score of the new strong classifier stage. The benefit of Recyling Cascade is the reduction of the number of the weak classifiers in the strong classifier stages and the reduction of the computation cost. The side-effect of Recyling Cascade is that the last stage of cascade can serve as an accurate strong classifier. Recyling & Retracting Cascade chooses a threshold after each weak classifier of the strong classifier is produced by Recyling Cascade to reject some negative sub-windows. To set these thresholds, it evaluates each score on the set of the positive examples and chooses the minimum score as the threshold so that all the positive examples in the set can pass all the weak classifiers.

These algorithms are evaluated on the standard MIT-CMU frontal face database [33], [8], which consists of 125 grayscale images containing 483 labeled frontal faces. If the detected rectangle and the ground-truth rectangle are at least 50 percent of overlap, we call the detected rectangle a correct detection. The number of average features per window is used to represent the computation cost. Fig. 18 reflects the computation cost of different algorithms (i.e., iCascade, Recyling Cascade and Retracting & Recyling Cascade) as a function of image location. The number of the average features used in a sliding window is accumulated to the center pixel of this sliding window. After detection, the value of each pixel is normalized to 0-255. The larger the value is, the greater the computation cost is, and the greater the probability that the face exists here is. It can be observed that Fig. 18(d) (i.e., iCascade) is much darker and sparser than Fig. 18(b) and (c). The darkness and sparisity imply that iCascade consumes less computation cost than the other two algorithms.

Fig. 19 shows the average number of features applied per window of different methods at different expected detection rates D_0. For example, when the detection rate is 0.98, iCascade averagely uses 5.95 features, wheras Fixed Cascade, Recyling Cascade and Reyling & Retracting Cascade use 22.84, 20.78 and 13.32, respectively. Fig. 20 shows the ROC of the different algorithms. The detection performance of different methods is no significant difference. From Fig. 19 and 20, we can conclude that iCascade has less computation cost with no loss of detection performance.



(a) Original image (b) Recyling response image

(c) Recyling & Retracting response image (d) iCascade response image

Fig. 18. The computation cost shown as a function of image location.

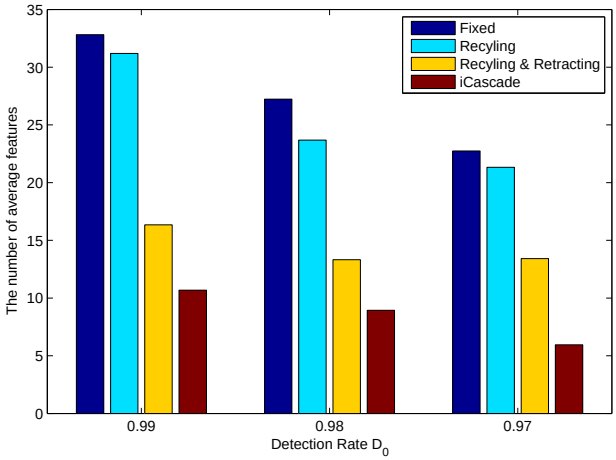


Fig. 19. Comparison of the computation cost between different algorithms

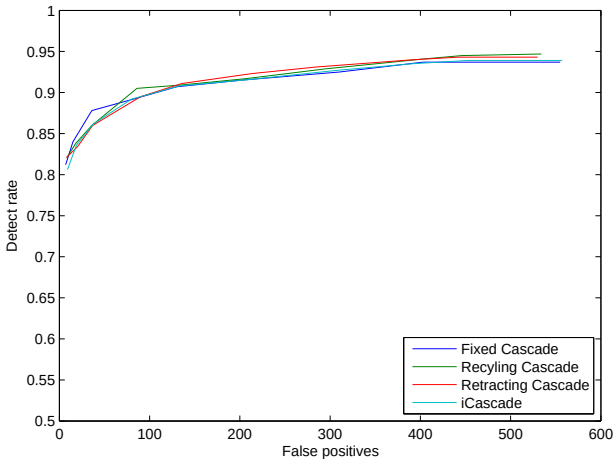


Fig. 20. ROC of different algorithms

VII. Conclusion

In this paper, we have proposed to design a one-stage cascade structure by partitioning a strong classifier into left and right parts. Moreover, we have proposed to design a multi-stage cascade structure by iteratively partitioning the right parts. Solid theories have been provided to guarantee the existence and uniqueness of the optimal partition point with the goal of minimizing computation cost of the designed cascade classifier. Decreasing phenomenon has been discovered and theoretically justified for efficiently searching the optimal solutions. In addition, we have presented an effective algorithm for learning the optimal threshold of each stage classifier.

Appendix A: Proof of Theorem 12

Proof. $\because \tilde{r}_2 = \arg\min_r f_2(r|\tilde{r}_1)$ and $r_2 = \arg\min_r f_2(r|r_1)$,

$\therefore \frac{df_2(\tilde{r}_2|\tilde{r}_1)}{d\tilde{r}_2} - \frac{df_2(r_2|r_1)}{dr_2} = 0$ is true.

Now investigate the value of $f_2'(\tilde{r}_2|\tilde{r}_1) - f_2'(r_2|r_1) = [p_1'(\tilde{r}_1)(\tilde{r}_1 - \tilde{r}_2 - c) - p_1'(r_1)(r_1 - r_2 - c)] + [p_1(\tilde{r}_1) - p_1(r_1)]$ if $\tilde{r}_2 > r_2$ is true:

$\because \tilde{r}_2 > r_2$ and $\tilde{r}_1 < r_1$ are assumed,

$\therefore 0 < p_1'(r_1) < p_1'(\tilde{r}_1), \tilde{r}_1 - \tilde{r}_2 - c < r_1 - r_2 - c < 0, 0 < p_1(\tilde{r}_1) < p_1(r_1)$,

$\therefore \; p_1'(\tilde{r}_1)(\tilde{r}_1 - \tilde{r}_2 - c) < p_1'(r_1)(r_1 - r_2 - c), p_1(\tilde{r}_1) < p_1(r_1)$.

$\therefore f_2'(\tilde{r}_2|\tilde{r}_1) - f_2'(r_2|r_1) < 0$.

This contradicts $f_2'(\tilde{r}_2|\tilde{r}_1) - f_2'(r_2|r_1) = 0$. Therefore, $\tilde{r}_2 > r_2$ is wrong and $\tilde{r}_2 \le r_2$ is true. □

Appendix B: Proof of Theorem 13

Proof. It is straightforward that $p_i(t_i)$ monotonically increases with t_i. Suppose we increase t_k in stage k from t_k^s to t_k^b with $t_k^s < t_k^b$ while the thresholds t_i in other stages (i.e., $i \neq k$) are fixed. Correspondingly, the rejection rate $p_k(t_k)$ grows from p_k^s to p_k^b and the computation cost f_S changes from $f_S(t_k^s)$ to $f_S(t_k^b)$. Theorem 13 can be proved if $f_S(t_k^s) - f_S(t_k^b) > 0$.

Define $f_k^L = \sum_{i=1}^{k}(r_i + ic)\left[\prod_{j=1}^{i}(1-p_{j-1})\right]p_i$, we have

$$
\begin{aligned}
f_S &= f_{k-1}^L + \sum_{i=k}^{S}(r_i + ic)\left[\prod_{j=1}^{i}(1-p_{j-1})\right]p_i \\
&\quad + (T + (S+1)c)\left[\prod_{j=1}^{S+1}(1-p_{j-1})\right] \\
&= \left[\prod_{j=1}^{k}(1-p_{j-1})\right]\left\{\sum_{i=k+1}^{S}(r_i + ic)\left[\prod_{j=k+1}^{i}(1-p_{j-1})\right]p_i \right. \\
&\quad \left. + (T + (S+1)c)\left[\prod_{j=k+1}^{S+1}(1-p_{j-1})\right] + r_k p_k\right\} + f_{k-1}^L.
\end{aligned}
$$

Because f_{k-1}^L is independent to the threshold parameter in stage k, so the difference between $f_S(t_k^s)$ and $f_S(t_k^b)$ is also

independent to f_{k-1}^L. Therefore,

$$\begin{aligned}
&f_S(t_k^s) - f_S(t_k^b) \\
&= \left[\prod_{j=1}^{k}(1-p_{j-1})\right]\left\{(r_k+kc)\left(p_k^s-p_k^b\right)\right. \\
&+(r_{k+1}+(k+1)c)\left[(1-p_k^s)p_{k+1}^s-(1-p_k^b)p_{k+1}^b\right] \\
&+\cdots \\
&+(r_S+Sc)\left[p_S^s\prod_{j=k+1}^{S}(1-p_{j-1}^s)-p_S^b\prod_{j=k+1}^{S}(1-p_{j-1}^b)\right] \\
&\left.+(T+(S+1)c)\left[\prod_{j=k+1}^{S+1}(1-p_{j-1}^s)-\prod_{j=k+1}^{S+1}(1-p_{j-1}^b)\right]\right\}.
\end{aligned} \tag{69}$$

Some items of (69) can be measured by using the fact that iCascade can reject all the true negative sub-windows in training data and the total rejection rate R is 1. The rejection rate R consists of the total rejection rate $R_{k-1} = \sum_{i=1}^{k-1}\left[\prod_{j=1}^{i}(1-p_{j-1})\right]p_i$ of the first $k-1$ stages and the total rejection rate $\tilde{R}_{k-1} = \sum_{i=k}^{S}\left[\prod_{j=1}^{i}(1-p_{j-1})\right]p_i + \prod_{j=1}^{S+1}(1-p_{j-1})$ of stages $k,\ldots,S$. That is,

$$\begin{aligned}
R &= \sum_{i=1}^{S}\left[\prod_{j=1}^{i}(1-p_{j-1})\right]p_i + \prod_{j=1}^{S+1}(1-p_{j-1}) \\
&= \sum_{i=1}^{k-1}\left[\prod_{j=1}^{i}(1-p_{j-1})\right]p_i + \sum_{i=k}^{S}\left[\prod_{j=1}^{i}(1-p_{j-1})\right]p_i \\
&\quad + \prod_{j=1}^{S+1}(1-p_{j-1}) \\
&= R_{k-1} + \tilde{R}_{k-1} = 1.
\end{aligned}$$

Because the total rejection rate R_{k-1} of the first $k-1$ stages does not change with t_k, so the total rejection rate $\tilde{R}_{k-1}$ of stages $k,\ldots,S$ is a constant η, no matter how t_k varies. So if we denote the rejection rates $\tilde{R}_{k-1}$ corresponding to p_k^s and p_k^b by $\tilde{R}_{k-1}(t_k^s)$ and $\tilde{R}_{k-1}(t_k^b)$, respectively, then $\tilde{R}_{k-1}(t_k^s) = \tilde{R}_{k-1}(t_k^b) = \eta$.

Therefore, when t_k grows form t_k^s to t_k^b, p_k will increase from p_k^s to p_k^b, the rejection rate in stage k will increase while the rejection rates in stages $k+1,...,S$ will decrease or not change (i.e., $\left[\prod_{j=1}^{i}(1-p_{j-1}^s)\right]p_i^s \geq \left[\prod_{j=1}^{i}(1-p_{j-1}^b)\right]p_i^b$ and $\prod_{j=1}^{S+1}(1-p_{j-1}^s) \geq \prod_{j=1}^{S+1}(1-p_{j-1}^b)$). So we have

$$\left[\prod_{j=k+1}^{i}(1-p_{j-1}^s)\right]p_i^s \geq \left[\prod_{j=k+1}^{i}(1-p_{j-1}^b)\right]p_i^b, \tag{70}$$

where $i = k+1,...,S$.

Based on (70), $f_S(t_k^s) - f_S(t_k^b)$ in (69) satisfies:

$$\begin{aligned}
&f_S(t_k^s) - f_S(t_k^b) \\
&> \left[\prod_{j=1}^{k}(1-p_{j-1})\right]\left\{r_k\left(p_k^s-p_k^b\right)\right. \\
&+r_k\left[(1-p_k^s)p_{k+1}^s-(1-p_k^b)p_{k+1}^b\right] \\
&+\cdots \\
&+r_k\left[p_S^s\prod_{j=k+1}^{S}(1-p_{j-1}^s)-p_S^b\prod_{j=k+1}^{S}(1-p_{j-1}^b)\right] \\
&\left.+r_k\left[\prod_{j=k+1}^{S+1}(1-p_{j-1}^s)-\prod_{j=k+1}^{S+1}(1-p_{j-1}^b)\right]\right\} \\
&= \left[\prod_{j=1}^{k}(1-p_{j-1})\right]r_k\left\{p_k^s+\sum_{i=k+1}^{S}p_i^s\prod_{j=k+1}^{i}(1-p_{j-1}^s)\right. \\
&+\prod_{j=k+1}^{S+1}(1-p_{j-1}^s)-\left[p_k^b+\sum_{i=k+1}^{S}p_i^b\prod_{j=k+1}^{i}(1-p_{j-1}^b)\right. \\
&\left.\left.+\prod_{j=k+1}^{S+1}(1-p_{j-1}^b)\right]\right\} \\
&= r_k\left\{\tilde{R}_{k-1}(t_k^s)-\tilde{R}_{k-1}(t_k^b)\right\} = r_k\{\eta-\eta\} = 0.
\end{aligned}$$

So $f_S(t_k^s) - f_S(t_k^b) > 0$ is proved. □

REFERENCES

[1] Ling Shao, Li Liu, and Xuelong Li, "Feature Learning for Image Classification Via Multiobjective Genetic Programming," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 7, pp. 1359-1371, 2014.

[2] Sakrapee Paisitkriangkrai, Chunhua Shen, and Anton van den Hengel, "A Scalable Stagewise Approach to Large-Margin Multiclass Loss-Based Boosting," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 5, pp. 1002-1013, 2014.

[3] Sakrapee Paisitkriangkrai, Chunhua Shen, Qinfeng Shi, and Anton van den Hengel, "RandomBoost: Simplified Multiclass Boosting Through Randomization," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 4, pp. 764-779, 2014.

[4] Wei Bian, Tianyi Zhou, A.M. Martinez, G. Baciu, and Dacheng Tao, "Minimizing Nearest Neighbor Classification Error for Nonparametric Dimension Reduction", *IEEE Trans. Neural Netw. Learning Syst.*, vol. 25, no. 8, pp. 1588-1594, 2014.

[5] Jifei Yu, Xinbo Gao, Dacheng Tao, Xuelong Li, and Kaibing Zhang, "A Unified Learning Framework for Single Image Super-Resolution," IEEE Trans. Neural Netw. Learning Syst., vol. 25, no. 4, pp. 780-792, 2014.

[6] P. Wang, C. Shen, N. Barnes, and H. Zheng, "Fast and Robust Object Detection Using Asymmetric Totally Corrective Boosting," *IEEE Trans. Neural Networks and Learning Systems*, vol. 23, no.1, pp. 33-46, 2012.

[7] Y. Pang, H. Yan, Y. Yuan, and K. Wang, "Robust CoHOG Feature Extraction in Human Centered Image/Video Management System," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 458-468, 2012.

[8] P. Viola and M. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision*, vol. 57, no. 2, pp. 137-154, 2004.

[9] C. Shen, P. Wang, S. Paisitkriangkrai, and A. Hengel, "Training Effective Node Classifiers for Cascade Classification," *Int'l J. Computer Vision*, vol. 103, no. 3, pp. 326-347, 2013.

[10] S. Brubaker, J. Wu, J. Sun, M. Mullin, and J. Regh, "On the Design of Cascades of Boosted Ensembles for Face Detection," *Int'l J. Computer Vision*, vol. 77, nos. 1-3, pp. 65-86, 2008.

[11] M. Saberian and N. Vasconcelos, "Boosting Classifier Cascades," *Proc. Advances in Neural Information Processing Systems*, 2010.

[12] L. Lefakis and F. Fleuret, "Joint Cascade Optimization Using a Product of Boosted Classifiers," *Proc. Advances in Neural Information Processing Systems*, 2010.

[13] M. Chen, Z. Xu, K. Weinberger, O. Chapelle, and D. Kedem, "Classifier Cascade for Minimizing Feature Evaluation Cost Minmin," *Proc. Int'l Conf. Artificial Intelligence and Statistics*, 2012.

[14] D. Benbouzid, R. Busa-Fekete, and B. Kégl, "Fast Classification Using Sparse Decision DAGs," *Proc. Int'l Conf. Machine Learning*, 2012.

[15] Z. Xu, M. Kusner, K. Weinberger, and M. Chen, "Cost-Sensitive Tree of Classifiers," *Proc. Int'l Conf. Machine Learning*, 2013.

[16] L. Bourdev and J. Brandt, "Robust Object Detection via Soft Cascade," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2005.

[17] R. Xiao, L. Zhu, and H. Zhang, "Boosting Chain Learning for Object Detection," *Proc. IEEE Int'l Conf. Computer Vision*, 2003.
[18] S. Li and Z. Zhang. "Floatboost Learning and Statistical Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1112-1123, 2004.
[19] H. Luo, "Optimization Design of Cascaded Classifiers," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2005.
[20] J. Wu, S. Brubaker, M. Mullin, and J. Rehg, "Fast Asymmetric Learning for Cascade Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 369-382, 2008.
[21] S. Brubaker, M. Mullin, and J. Rehg, "Towards Optimal Training of Cascaded Detectors," *Proc. European Conf. on Computer Vision*, 2006.
[22] M. Pham, V. Hoang, and T. Cham, "Detection with Multi-exit Asymmetric Boosting," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2008.
[23] X. Chen and A. Yuille, "A Time-Efficient Cascade for Real-Time Object Detection: With Applications for the Visually Impaired," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2005.
[24] J. Sochman and J. Matas, "WaldBoost-Learning for Time Constrained Sequential Detection," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2005.
[25] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2006.
[26] X. Wang, T. Han, and S. Yan, "An HOG-LBP Human Detector With Partial Occlusion Handing," *Proc. IEEE Int'l Conf. Computer Vision*, 2009.
[27] P. Dollár, Z. Tu, P. Perona, and S. Belongie. "Integral Channel Features," *Proc. British Machine Vision Conf.*, 2009.
[28] R. Benenson, M. Mathias, R. Timofte, and L. Gool, "Pedestrian Detection at 100 Frames per Second," *Proc. IEEE Int'l Conf. Computer Vision*, 2012.
[29] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743-761, 2012.
[30] B. Jun, I. Choi, and D. Kim, "Local Transform Features and Hybridization for Accurate Face and Human Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1423-1436, 2013.
[31] G. Gualdi, A. Prati, and R. Cucchiara, "Multi-Stage Particle Windows for Fast and Accurate Object Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1589-1604, 2012.
[32] C. Zhang and P. Viola, "Multi-Instance Pruning for Learning Efficient Cascade Detectors," *Proc. Advances in Neural Information Processing Systems*, 2007.
[33] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 22-38, 1998.